USO DE MEDIDAS DE COMPLEXIDADE EM SELEÇÃO DE ATRIBUTOS

USE OF DATA COMPLEXITY MEASURES IN FEATURE SELECTION

LUCAS CHESINI OKIMOTO

Lucas Chesini Okimoto: *Use of Data Complexity Measures in Feature Selection*, Master's Dissertation presented to the Institute of Science and Technology - Universidade Federal de São Paulo - UNIFESP, as part of the activities to obtain the title of Master in Computer Science, Advisor: Profa. Dra. Ana Carolina Lorena, July 2018

TO SANTIAGO,
my nephew.

I wish you a world
free of demons
and full of light.

"We may have found that perspective just in time, just as our technology threatens the habitability of our world. Whatever the reason we first mustered the Apollo program, however mired it was in Cold War nationalism and the instruments of death, the inescapable recognition of the unity and fragility of the Earth is its clear and luminous dividend, the unexpected final gift of Apollo. What began in deadly competition has helped us to see that global cooperation is the essential precondition for our survival.

Travel is broadening.

It's time to hit the road again".

— Carl Sagan. "Pale Blue Dot: A vision of the human future in space." Ballantine Books (September 1997) p. 171

ACKNOWLEDGEMENTS

To my family, for unconditional support. For the creation, education and values passed. People who did not have half the opportunities they were able to give me and I dream, someday, be able to repay in double for what they did for me.

To my advisors, Ana Carolina Lorena and Andre C P L F de Carvalho. To Andre for accepting and directing me. And to Ana, for the long hours of support and patience and, simply, for the incredible person that she is. I will be eternally grateful to both for all the dedication and knowledge that changed my life.

To my friends Bruno, Daniel, Leonardo, Matheus, Rafael and Rhander, with whom I could always count on. Amazing people who taught me a lot over the years.

To professor Jurandy, for the tireless lessons and sincere desire in teaching.

To all professors of UNIFESP, for the knowledge granted to me.

To Jefferson de Souza, for the inexplicable meeting on my first day of masters. For strong advices that shaped this work.

To Borja Seijo-Pardo and Verónica Bolón-Canedo to the datasets used in this work and Ricardo Manhães Savii for the complexity measures' Python code.

To GPAM, Machine Learning Research Group, where incredible people have passed and many others will come.

To the Universidade Federal de São Paulo (UNIFESP), for the structure and technical support.

To CAPES, for the financial support.

To God, for the life.

ABSTRACT

Feature selection is an important pre-processing step usually mandatory in data analysis by Machine Learning techniques. Its objective is to reduce data dimensionality by removing irrelevant and redundant features from a dataset. In this work we evaluate the use of complexity measures of classification problems in feature selection (FS). These descriptors allow estimating the intrinsic difficulty of a classification problem by regarding on characteristics of the dataset available for learning. We propose a combined univariate-multivariate FS technique which employs two of the complexity measures: Fisher's maximum discriminant ratio and intra-extra class distances. The results are promising and reveal that the complexity measures are indeed suitable for estimating feature importance in classification datasets. Large reductions in the numbers of features were obtained, while preserving, in general, the predictive accuracy of two strong classification techniques: Support Vector Machines and Random Forests.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AUC Area Under the ROC Curve

CAPES Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

CBFS Consistency Based Feature Selection

CFS Correlation Based Feature Selection

CR Cumulative Relevance

DT Decision Tree

F1 Maximum Fisher's Discriminating Ratio

F2 Volume of the Overlapping Region

F3 Maximum Individual Feature Efficiency

F4 Collective Feature Efficiency

FCBF Fast Correlation Based Feature Selection

FP False Positive

FS Feature Selection

GA Genetic Algorithm

kNN k Nearest Neighborhood

L1 Sum of the Error Distance by Linear Programming

L2 Error Rate of Linear Classifier

L3 Non Linearity of Linear Classifier

LS Laplacian Score

ML Machine Learning

mRmR Minimum-Redundancy-Maximum-Relevancy

MST Minimum Spanning Tree

N1 Fraction of Borderline Points

N2 Ratio of Intra/Extra Class Nearest Neighbor Distance

N3 Error Rate of the Nearest Neighbor Classifier

N4 Non Linearity of the Nearest Neighbor Classifier

NP Nondeterministic Polynomial time

PCA Principal Component Analysis

RE Representation Entropy

RF Random Forest

ROC Receiver Operating Characteristic

SVM Support Vector Machines

T1 Fraction of Hyperspheres Covering Data

TP True Positive

UMFS Univariate-Multivariate Feature Selection

1 INTRODUCTION

In the past years, the advance of digital technologies and their widespread usage around the globe have generated an unprecedented amount of data in a speed beyond the human capacity to process it. In order to deal with the challenge of analyzing such data, many knowledge extraction and data-preprocessing techniques have been proposed [21]. Among them are the Machine Learning (ML) techniques, which are able to extract predictive and descriptive models from past known data [41]. This work deals with a pre-processing step usually applied in ML studies aimed to reduce data dimensionality: feature selection (FS) [37].

FS is applied to the training learning dataset in ML. This dataset is composed of n examples, described by m input features (attributes or characteristics). Here we deal with standard classification problems, so that each example is labeled into one specific class. FS techniques try to find a subset of the original input features with reduced size $m' < m$. This subset should discard irrelevant and redundant input features from the dataset. In contrast to other dimensionality reduction techniques like those based on data projection (e.g. PCA) [34], FS does not alter the original representation of the variables, but merely selects a subset of them. Therefore, they preserve the original semantics of the variables, maintaining data interpretability. The underlying idea of FS is that by using fewer input features on data [24]: the learning process can become faster; models can generalize better; and the obtained results can be simpler and easier to understand. Moreover, this allows a better comprehension of which are the most important features or characteristics that are related to a given problem.

Finding the optimal number and subset of features that lead to a higher classification performance is not an easy-to-do task. The main reasons are [37]:

- We often do not have a precise idea of which features are relevant to a given problem, so we tend to add as much information as we can gather when a dataset is built.
- Data is often collected for a variety of situations and purposes. For instance, it can be part of a routine like a demographic census. In other cases, data may be merged from multiple sources. In these situations, it is likely to include irrelevant and redundant features that undercover relevant features and increase data dimensionality.
- Having more features often means the need for more instances since we need to ensure the statistical variability between patterns from different classes [54]. But the training dataset has a limited number of training instances, and it can be hard - in some cases even unfeasible - to get new examples.

In this work, we hypothesize that the presence of irrelevant features in a dataset may increase the complexity involved in the classification problem solution. We thereby investigate how measures devoted to estimating the complexity of a classification problem can be used to guide the discard of irrelevant features in a training

dataset. These measures are geometric and statistic descriptors extracted from the training datasets that allow one to estimate how difficult the solution of the classification problem shall be [30]. There are a variety of such measures in the literature, and we investigate which can be more effective in the FS process.

Some of the measures have a univariate nature, evaluating each feature individually. Others are more suitable for multivariate data. A univariate analysis allows ranking the features based on a given complexity measure value at a lower computational cost. But this disregards interactions between the features, which may be taken into account in a multivariate analysis. Whilst costly, a multivariate analysis can reveal important relationships between the input variables. Therefore, we propose to join two complexity measures in a novel FS technique: the maximum Fisher's discriminant ratio (univariate) and the intra-extra class distances (multivariate). The choice of each of these constituent measures is based on experiments performed on synthetic datasets for which the identity of the relevant features is known. Next, the results of the FS technique are evaluated on more datasets by comparing the predictive performance achieved by using all features against the ones recorded on the reduced subsets.

1.1 MOTIVATION

Machine Learning (ML) algorithms use an inductive principle in order to generalize from known data. They can be roughly divided into three major groups according to their learning strategy: supervised, semi-supervised and unsupervised [21].

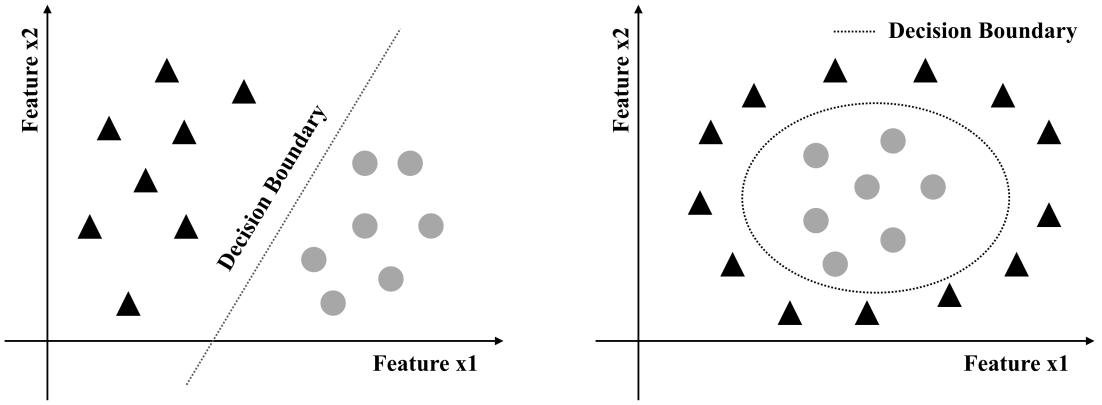
In supervised learning, the idea is that there is an external supervisor, who knows the desired outcome for each training instance. It uses this knowledge to label the data. The learning technique must find the hidden relationship between the input features and the recorded labels. Supervised ML can also be divided by the label data type: i) classification for discrete outputs and ii) regression for continuous outputs. This work is focused on supervised classification problems.

In unsupervised ML, no label is added to the training data. Herein, these algorithms search for relationships able to describe the data. It can be used, for instance, to find groups of similar objects. It can also be divided into three groups: i) clustering, in which data are grouped by similarity, ii) association, which finds frequent patterns on a dataset, and iii) summarization, which finds a simple description of the dataset.

In semi-supervised learning, one has both types of data: labeled and unlabeled. Usually unlabeled data is available in a larger amount, since the labeling process by a domain specialist can be considered costly. Therefore, in this learning paradigm, the unlabeled data is used as additional information in the learning process [14, 6].

Classification is the basic problem in pattern recognition where one wants to classify a given instance into one out of n_c known classes. An instance is represented by a set of features, presumably, containing information that may distinguish instances among different classes. They are often represented in "attribute-value" format. The training dataset X contains n instances represented by m features plus the class label. The goal is to induce a classifier to predict the class of unseen instances.

Figure 1 presents examples of two classification datasets with two input features and two classes. The dataset in Figure 1a is linearly separable, and a linear boundary is able to separate the examples from the two classes. Figure 1b, on the other hand,



(a) Linear boundary

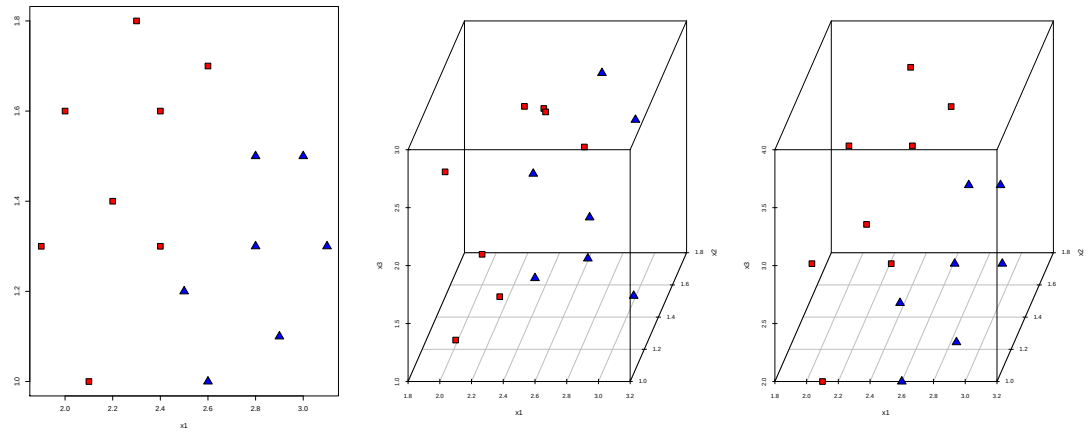
(b) Non-linear boundary

Figure 1: Linear versus non-linear classification boundary

illustrates a dataset for which a more complex non-linear separation boundary is needed. Considering that the decision boundary will be highly dependent on data distribution and that it was empirically observed that there is no ML technique which can perform well for all classification problems [56], some work has been devoted to understanding what makes a classification problem complex. In this context, Ho and Basu [30] proposed a set of measures that extract descriptors from data, in a meta-analysis. These measures evaluate different characteristics of the data and can be grouped as: i) feature overlapping measures; (ii) class separability measures; and (iii) measures of geometry, topology and density of manifolds. Therein, most of these indexes shall be able to reflect that the classification problem in Figure 1a is simpler than the classification problem in Figure 1b.

According to [38], the complexity of classification boundary can be related to its Kolmogorov complexity [40]. It can be thereby measured by the size of the smallest algorithm able to describe the relationships between the data. If there is some regularity in the data, a compact description can be obtained. But in the worst case, all the objects along with their labels have to be listed. In practice, the Kolmogorov complexity is not computed but rather approximations are made, such as those from the complexity measures of [30].

In this work, we use these complexity measures in FS. The idea is that they can be used to evaluate the importance of the input features in a classification dataset and reveal the presence of irrelevant features. Take, for instance, the linearly separable classification dataset from Figure 2a. Figure 2b shows the same classification problem after a random input feature is added to the original dataset. It is possible to notice that the data relationships are harmed by this feature, making the classification problem more complex (the classes now overlap). This is the main motivation of applying the complexity measures in the identification of irrelevant features in a classification dataset. We do not expect to remove redundant features in this process, since they shall not modify the underlying problem complexity. For instance, Figure 2c introduces the second feature multiplied by two into the dataset from Figure 2a. It is noticeable that the problem complexity is not affected in this case and the problem remains linearly separable.



(a) Initial solution (b) Addition of a random feature. (c) Addition of a redundant feature.

Figure 2: Effects of the addition of random irrelevant (b) versus the addition of redundant (c) features in a classification problem (a).

According to Liu and Motoda [37], multiple criteria can be used to evaluate the importance of a feature or a subset of features in a classification dataset. They define five broad groups of feature importance measures: consistency, dependency, distance, information and precision. Consistency measures try to select features that allow building a consistent hypothesis from data. Dependency measures are based on the correlation or association of the features between each other and/or to the class feature. Distance measures try to capture the power of the features in discriminating the classes, usually by regarding on neighborhood information. Information measures quantify the uncertainty associated with the use or removal of one or more of the features. Finally, precision measures consider the classification performance achieved by using some features. Here we evaluate the use of the complexity descriptors as a sixth feature importance category.

The **hypothesis** of this work is that the data complexity measures can be used to identify irrelevant features in a dataset and thereby support FS.

1.2 OBJECTIVES

The main **objective** of this work is to use the complexity measures for evaluating the importance of the input features of a classification dataset in FS. The specific objetives are:

- Define and evaluate strategies for incorporating the complexity measures in the FS process;
- Determine which complexity measures are more suitable for FS;
- Propose a FS algorithm which is guided by one or more of the complexity measures values.

1.3 WORK ORGANIZATION

This work is organized as follows:

- Chapter 2 reviews feature selection literature;
- Chapter 3 presents all complexity measures used in this work;
- Chapter 4 presents our proposal;
- Chapter 5 presents the experiments done in order to evaluate our FS algorithm;
- Chapter 5 concludes this work.

2 FEATURE SELECTION

In the context of knowledge discovery and high-dimensional data, feature selection (FS) plays an important role. Liu and Motoda [37] define FS as "*[...] a process that chooses an optimal subset of features according to a certain criterion*". The selection of a subset of features is determined by the identification and selection of only relevant and non-redundant features. The high-dimensional datasets, also called small-n-large-m datasets ($m >> n$), are one of the biggest challenges for researchers. In this type of dataset the number of features is much larger than the sample size, and usually data are sparse. This means that only a few features really affect the response variable. For these two reasons feature selection here is a task on which researchers focus most.

First we need to define what is a relevant feature for a given problem. Blum and Langley [8] consider a feature f_i relevant if there is a pair of examples $\mathbf{x}_A$ and $\mathbf{x}_B$ in the instance space that differ from each other in respect to the classes by using such feature. The definition of relevance according to the ReliefF [35] FS algorithm is relative to how discriminative a feature is to distinguish examples from the same class from those from different classes.

It is also important to define redundancy, which is usually explained in terms of correlations between features [57]. Two features are redundant if their values are completely correlated, therefore, only one of them would suffice to describe the data. It is important to notice that not all feature importance criteria are able to assess feature redundancy.

Herein, we define as relevant those features that reduce the complexity of a classification problem, since all ML technique are extremely data dependent and biased by the complexity of the data. We use the complexity measures proposed by Ho and Basu [30] to assess feature relevance according to the previous definition.

2.1 FORMAL DEFINITION OF FEATURE SELECTION

Let X be a dataset containing n pairs $(\mathbf{x}_i, y_i)$ of examples, where $\mathbf{x}_i \in X$ is a data item described by m input features and $y_i \in \{1, \ldots, n_c\}$ corresponds to its class, where n_c is the total number of classes. The objective of FS in classification problems is to select a minimum feature subset $m' < m$ so that the achievable performance using this subset is similar or superior to that which can be obtained by using all m original features [25].

2.1.1 *Univariate vs Multivariate Feature Selection*

FS can be divided in two main approaches: univariate and multivariate. In the former, features are evaluated independently from each other. Usually this is done by considering the intrinsic properties of the data and the outcome is a ranking of the features according to their relevance. As they evaluate a single feature at a time,

univariate methods are unable to identify interactions and redundancies between features. Indeed, features of similar relevance will be ranked next to each other.

In the latter, a subset of features is evaluated using a certain search strategy. Herewith, it is able to identify relevancy and can consider dependencies between features too. Please note that these dependencies are not necessarily related to feature redundancy, but can be due to joint relevance instead. In this strategy it is necessary to consider possible combinations of the input features. The search space grows exponentially when m increases due to the relationship $S = 2^m$, where S is the size of a search space with respect to m, and 2 represents two possible choices: whether to select or not each feature. In order to go through the space of possibilities, a search strategy must be employed. According to Blum and Langley [8], the nature of the search process is four-fold: i) search direction, ii) search strategy, iii) feature importance, and iv) stopping criterion.

2.1.2 *Search Direction*

Figure 3 illustrates a search space for a problem with three input features. Selected features are colored in black. At the two ends of the figure lies two extreme subsets: one is empty without any feature and the other is full with all three features selected.

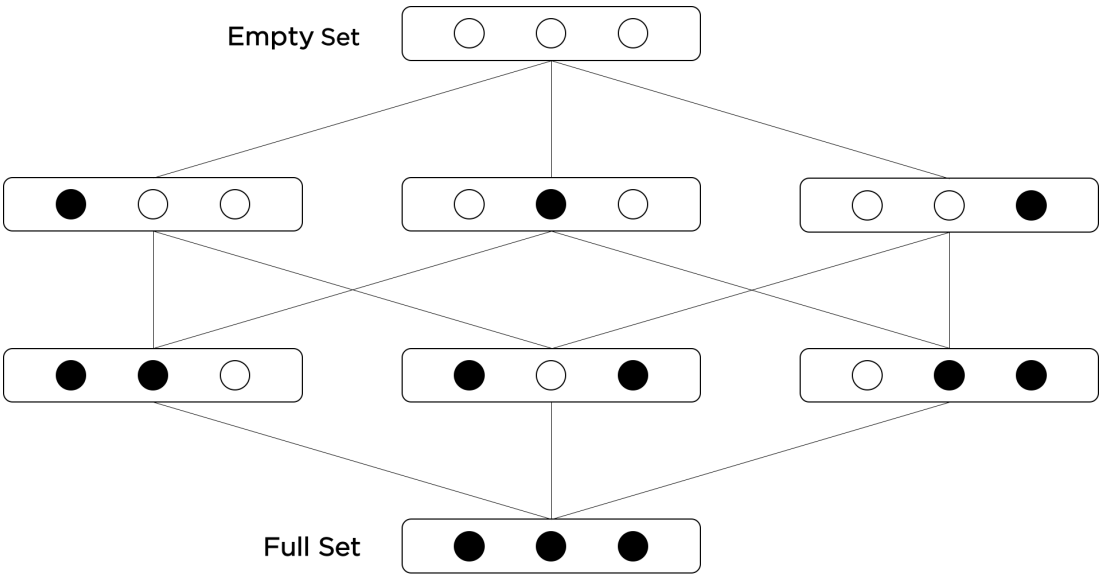


Figure 3: The search space for a problem with three input features [37]. Dark dots represent chosen features.

If there is no prior knowledge about where the optimal feature subset is in the search space, it would make no difference whether to start the search from an empty set or a full set [37]. On average, for a large number of problems, both directions would find the optimal feature subset at the same speed. Search directions are closely related to the feature subset generation process. One direction is to grow a feature subset from an empty set (e.g., Forward Selection) and the other direction corresponds to gradually remove the least important features from the full set (e.g., Backward Selection).

- Forward Selection: it begins with an empty set and features are added into the selected subset F successively. At the first round, the best feature is added into F based on some evaluation function. Next, all unselected features are tested with the existing feature and the feature which returns the best combination according to the evaluation function is added into F and so on. Therefore, the selection of each of the next features will be highly dependent on the first feature added. The feature subset grows until it reaches a given stop criterion (e. g. no significant improvement on prediction performance occur). The order in which each feature is added to F can be used to rank the selected features too. Given a threshold (say θ), we can simply choose the first θ features in the ranked list.

- Backward Selection: it begins with the full set and features are removed iteratively. At each iteration, the least important feature is removed based on some evaluation function. The feature subset shrinks until it reaches a given stop criterion. A rank list can also be established according to the order the features are removed (important features would be removed later).

Forward selection complements backward selection because on the latter the majority of the search occurs near the full feature subset instead of near the empty feature subset, and the other way around in the former. Both strategies have their advantages and disadvantages. Starting with the full set of features can benefit from considering all information provided, and the selection of the least relevant feature would be, ideally, more precise than on forward selection. On the other hand, forward selection may minimize effects of noise presence in the FS process, since irrelevant features can be discarded in earlier steps.

It is also possible to perform other types of search, such as:

- Bidirectional Selection: it starts the search in both directions, i.e., forward selection starts from the empty set and backward selection from the full set, converging to the middle of the search space on average. They can stop in two cases: i) when either search finds the best feature subset; or ii) when both searches reach each other.

- Randomized Selection: it starts the search in a random direction. The addition or removal of a feature is also done at random. The main advantage of randomized selection is that it tries to avoid trapping into local optima.

Using any of the four types of search directions, it is possible to design FS search for different needs. Sometimes, we may want a quick solution to get an overview of the problem and, sometimes we need a deeper look no matter the cost of getting it. In the majority of the cases, we may want a FS procedure that improves prediction performance. In combination with a variety of search strategies, it is possible to design a FS algorithm that fits to the problem at hand.

2.1.3 *Search Strategy*

Another important decision to make in FS problems is the trade-off between optimality and execution time [19]. Without taking proper care, algorithms that converge

fast will tend to stop on local minimum and algorithms that can potentially find the global minimum will almost go through the entire search space. Thus, the search strategy employed is the dimmer between the two and can be divided in three groups: i) exhaustive/complete, ii) heuristic, and iii) randomized search.

Since the search space is in the order of $O(2^m)$ and FS problem becomes NP-hard for a medium-to-high number of features [36], an exhaustive search is usually impractical [8]. However, this is the only search strategy that can guarantee optimality. In order to get a solution in a reasonable amount of time, we have to sacrifice optimality of the selected subsets. A heuristic can be used to guide the search into the search space. It will not go through the entire search space, hence it risks losing optimal solutions. Hill-Climbing and Best-First are some of the most used heuristic search strategies [53, 36]. For instance, the Best-first strategy selects the most promising feature subset that has not been considered yet.

A common drawback of heuristics is that they tend to get trapped into local minima. Randomized search strategies can escape from local minima by using randomness to eventually accept worse states. The idea is to move into regions where the search can be more fruitful in finding the global minimum. In this kind of search, the next feature subset is set at random so it can shrink and expand at each round, and do not follow a straight direction. But they are usually not stable, meaning that every execution will produce a different result. Genetic Algorithm and Simulated Annealing are two of the most used algorithms in this type of search strategy [13, 37].

2.1.4 *Feature Importance*

The optimal feature subset will be always relative to the evaluation function employed. Hence, the "most promising" feature can vary for different evaluation functions. Liu and Motoda [37] define a feature as relevant as "*[...] the feature that if it is removed, the measure of the remaining features will deteriorate [...]*". The authors define five broad measures to determine feature relevancy: accuracy, consistency, information, distance, and dependence.

Dependence measures quantify the ability to predict the value of one variable from the value of other variable. For instance, the variable to be predicted can be the class. If the correlation of feature f_i with class C is higher than the correlation of feature f_j with C, then the feature f_i is preferred over f_j. A slight variation of this is determine the dependence between different input features, revealing their redundancy. The CFS (Correlation based Feature Selection) [26] algorithm is a representative of this category.

Distance measures take into account the distances between instances for discriminating the classes. The larger the distance between two or more classes using a given feature, the easier and more relevant is the feature to discriminate between different classes. If this distance is null, then the feature is irrelevant. ReliefF [35] is an example of FS algorithm which employs a distance measure for evaluating the features. Laplacian Score (LS) [29] is another kind of distance measure. The premise is that nearby examples are possibly related, so they have higher chances of being from the same class.

Information measures evaluate the uncertainty related to a feature. The information gain of a feature f_i can be computed by the difference in uncertainty before and

after using this feature. A feature f_i is more relevant than another feature f_j if the information gain using f_i is greater than that from using feature f_j. Information Gain (IG) [27] and Representation Entropy (RE) [42] are examples of this kind of measure.

Consistency Measures rely on min-features bias, which prefers consistent hypotheses definable over as few features as possible [1]. The Consistency Based Feature Selection (CBFS) [27] technique is a representative that employs a consistency feature importance measure.

Finally, accuracy measures use a classifier to determine feature importance. The best feature is the one that maximizes classification accuracy. They can also be called wrappers methods [55], as discussed next. This kind of method is biased by the classifier used on FS, hence it cannot guarantee the same performance when using a different algorithm for classification.

In [8], the authors extend the definition of relevancy as a measure of complexity, focusing in how complex is a function rather than caring about which are exactly the relevant features. In fact, in this work we adopt the same concept and employ the complexity measures proposed by Ho and Basu [30] in feature importance evaluation. The idea is that the presence of irrelevant features will make the classification problem more complex.

2.1.5 *Stopping Criterion*

The search for a feature subset in FS algorithms can be stopped in various ways. If the optimal number of features is known in advance, one may stop when a subset of this size is found. A threshold on the number of features to be selected can also be established a priori. For example, one may want to reduce the original number of features by half, as done in [42]. It is also possible to monitor the values of the feature importance measures and decide upon a stopping criterion. In [4], the authors keep the features for which the importance values are in between two variances from the mean. In [39], the authors use a threshold based on the largest gap between two consecutive feature importance values. Finally, some algorithms may have a specific number of iterations to be run. In [23], the authors compare results achieved after a limited period of runtime (runtimes of 1, 8, 16, 24, 72, 168 and 240 h). This is also the case of Genetic Algorithms, which are usually stopped after a given number of generations is executed.

In this work, we used two approaches: one for the univariate case and another for multivariate setup. In the former, we monitor the feature importance values and selected the top-m' ranked features based on their cumulative relevance (CR) as proposed in [22]. We first calculate the complexity measure for every feature and rank them by assigning the most relevant the first position. We than compute CR as follows. Let c_i be the complexity measure value for feature f_i and

$$CR_i = \sum_{i=1}^{m} c_i \tag{1}$$

where m is the total number of features. We then return the number of features that add up to γ percent of CR, where γ is some confidence level (e. g. $\gamma = 0.95$), similar to Pareto's distribution [44].

In the multivariate setup, we decided to set a threshold based on the number of features m, as defined and used in [10]:

- if $m < 10$, select 75% of the features;
- if $10 \leqslant m < 75$, select 40% of the features;
- if $75 \leqslant m < 100$, select 10% of the features;
- if $m \geqslant 100$, select 3% of the features.

These criteria allow to keep a low number of features even for datasets of a very high dimensionality. Since the multivariate FS is more costly, this also allows to reduce our overall running time.

2.2 FEATURE SELECTION METHODS

There are three main approaches in which FS can be performed: Filter, Wrapper and Embedded [36]. Filter methods have, in general, the lowest computational cost and are largely applied on high-dimensional datasets due to their simplicity. They usually consist on simple descriptors used to evaluate the features from the dataset. Features achieving a score bellow a given threshold are eliminated. Wrapper methods analyze features in a combinatorial way and select a subset of features that maximize the predictive power of a given classifier. In order to achieve that, the classifier is wrapped in the search algorithm. For this reason, the computational cost of wrapper methods is usually expensive. In embedded methods, the classifier itself, by inducing the predictive model, also performs FS.

It is also possible to hybridize these methods, as in [5, 13, 48], combining their advantages. For instance, one can use filter methods of low computational cost to pre-select relevant features. This feature subset is then refined by a wrapper method, with a lower cost than that would be obtained when analyzing the original dataset.

2.2.1 *Filter Methods*

Filter techniques are the most used FS method in high dimensional datasets due to their simplicity and low computational cost [12]. These methods are, in general, univariate and evaluate intrinsic properties of the data. Features are ranked in descendant order and a threshold is applied to the obtained scores so that the top ranked features are selected and can be then used to induce a classification model (as shown in Figure 4).

A common drawback of this method is that features with high correlation will have similar scores and will be ranked closely. Therefore, redundant features may be selected. Fisher Score [13], F-test [7], and Information Gain [16] are the most used univariate filter techniques. On the other hand, ReliefF [35], Correlation Based Feature Selection (CFS) [26], Minimum-Redundancy-Maximum-Relevancy (mRmR) [18], and Fast Correlation Based Feature Selection (FCBF) [57] are multivariate filter techniques that have been applied in the literature. They all use different mathematical approaches to evaluate the importance of the features and to choose the number of

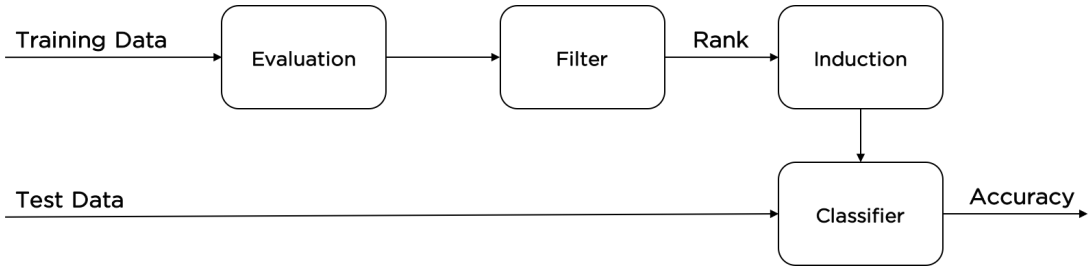


Figure 4: Filter methods workflow. On the top left corner, a training set is given as input to a filter method that outputs a rank of features. Features above a threshold are selected and the others are eliminated. Afterwards, a classifier is trained on the selected features and validated on the test set.

features to be output. Our FS technique can also be regarded as a filter, as described in Chapter 4.

Another relevant point is that filter methods are independent of the classification algorithm. Therefore, the obtained results can be considered more general and unbiased towards a specific classification technique.

2.2.2 *Wrapper Methods*

In this setup, a search procedure in the space of possible feature subsets is defined, and various subsets of features are generated and evaluated. The evaluation of a specific subset of features is obtained by training and testing (in a validation set) a specific classification model, rendering this approach tailored to a specific classification model. Thus, the subset of features is the one that maximizes the accuracy performance or another performance metric of the classifier [48, 13]. An illustration of the wrapper work-flow is shown on Figure 5.

Wrapper techniques are biased by the classification algorithm because they "wrap" the classifier into FS. In this way, they can evaluate and identify both relevant and redundant features, but are computational intensive. The predictor works as a "black box" in the search process, as follows:

- Step 1: determining a subset of features according to a search strategy;
- Step 2: evaluating the selected subset of features by the performance of the classifier on a validation set;
- Step 3: repeating Step 1 and Step 2 until a desired quality level is reached.

Wrapper methods have been used in the literature with a combination of different search strategies. For instance, hill-climbing and Best-First are well-known algorithms for sequential search. On randomized/heuristic search side, Genetic Algorithms is one of the most used search algorithms, as already mentioned.

The main drawback of wrapper methods is that they are computationally intensive. They have also a higher chance to overfit for a given classification model [53]. But, in general, the predictive performance obtained by the feature subset output by a wrapper technique tends to offer the higher predictive performance for the considered classification technique.

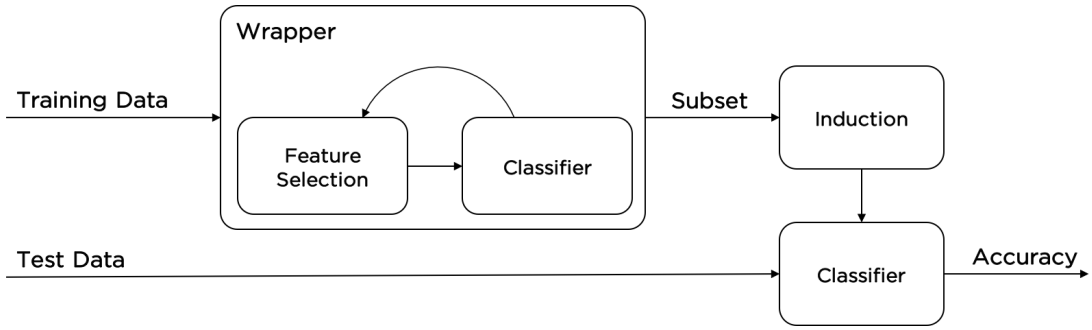


Figure 5: Wrapper methods workflow. The big box on the top is known as "black box". Inside it, the algorithm iteratively searches for a combination of features that maximizes the classification performance.

2.2.3 *Embedded Methods*

Proposed to bridge the gap between filter and wrapper techniques, embedded techniques try to explore the accuracy of the wrapper and the efficiency of the filter model. It embeds prediction and FS simultaneously. The search for an optimal feature subset is also part of the classifier construction process. Embedded methods have the advantage that they include the interaction with the classification model, while at the same time being far less computationally intensive than wrapper methods. Nonetheless, this FS process is intrinsic to a few classification techniques only, such as Decision Trees (DT) [47].

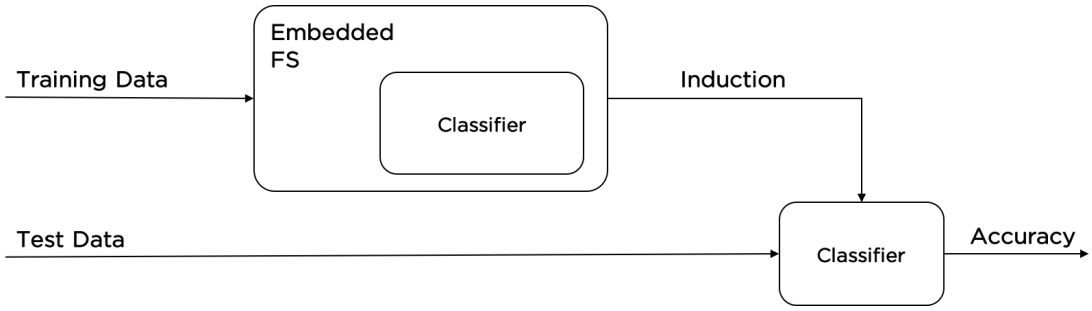


Figure 6: Embedded methods workflow. Feature selection runs simultaneously with classification. The output is the induction itself.

Figure 6 illustrates the embedded FS workflow. Indeed, the predictor itself outputs as an additional result the subset of features it identified as important during its induction process. Again, the obtained subset of features is biased towards the specific classification algorithm used and may not generalize well to other classification techniques.

3

COMPLEXITY MEASURES

Along the 1990s, many studies have been done to understand what can improve the predictive power of a variety of classification models. From the observation that every ML algorithm is very data dependent, some metadata started to be extracted from the classification problems in an attempt to understand what makes a classifier to perform better under certain circumstances and poorly under others. In [30], the authors proposed several measures to evaluate the complexity of a classification problem. They reveal the expected complexity of the classification boundary that must be used to separate the training data points into their classes. The proposed measures were divided in three main groups: i) feature overlapping, ii) class separability and, iii) geometry, topology and density of manifolds. They are mainly geometrical and statistical measures that calculate, for instance, how discriminant a feature is regarding feature overlapping, or how complex is the area in the frontier of the classes.

The rest of this work will refer to the complexity measures by the acronyms shown in Table 1. Some standardizations were applied to some measures in order to make all of them vary in the $[0, 1]$ interval and so that higher values (closer to 1) denote more complex problems.

Table 1: Identification of Complexity Measures

Category	Measure	Acronym
Feature Overlapping	Maximum Fisher's Discriminating Ratio	F1
	Volume of the Overlapping Region	F2
	Maximum Individual Feature Efficiency	F3
	Collective Feature Efficiency	F4
Separability of Classes	Sum of the Error Distance by Linear Programming	L1
	Error Rate of Linear Classifier	L2
	Fraction of Borderline Points	N1
	Ratio of Intra/Extra Class Nearest Neighbor Distance	N2
	Error Rate of the Nearest Neighbor Classifier	N3
Geometry, Topology, Density	Non Linearity of Linear Classifier	L3
	Non Linearity of the Nearest Neighbor Classifier	N4
	Fraction of Hyperspheres Covering Data	T1

3.1 MEASURES OF FEATURE OVERLAPPING

These measures quantify whether there is at least one input feature able to discriminate the training data. All measures presented require features with numeric values.

3.1.1 *Maximum Fisher's Discriminant Ratio (F1)*

Maximum Fisher's Discriminant Ratio (F1) is a measure that is already largely applied on FS [33]. It assesses the overlap of the value of a features in different classes according to the Equation 2:

$$F1 = \max_{i=1}^{m} \frac{\sum_{j=1}^{k} n_{c_j} (\mu_{c_j}^{f_i} - \mu^{f_i})^2}{\sum_{j=1}^{k} \sum_{l=1}^{n_{c_j}} (x_{li}^{j} - \mu_{c_j}^{f_i})^2}, \tag{2}$$

where n_{c_j} is the number of examples in class c_j, μ^{f_i} is the average of the f_i values, despite of the classes, and x_{li}^{j} denotes the individual values of the feature f_i for examples from class c_j. It takes the maximum discriminant ratio among all input features.

In this work, in order to standardize the results a mathematical manipulation was done. Our results were given according to

$$F1' = \frac{1}{1+F1} \tag{3}$$

where F1' $\in$ [0,1]. If at least one feature is highly discriminative, the F1' value will be closer to zero. Figure 7 presents an example. Therein, feature f_1 allows data to be perfectly discriminated, whilst feature f_2 shows a high overlapping. Lower F1' values are obtained for simpler problems, for which a hyperplane perpendicular to one of the feature's axis can separate the classes.

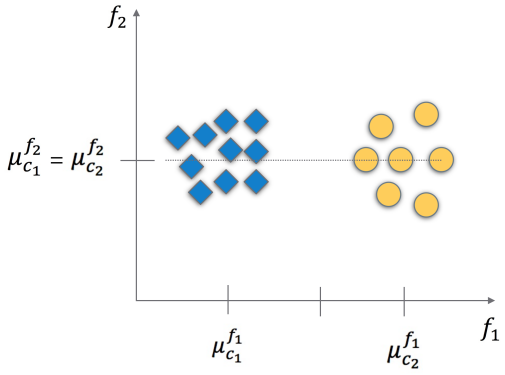


Figure 7: Feature f_1 is discriminative whilst f_2 is not due to a overlap of means.

The numerator goes through the classes. Since the discriminant ratio must be computed for all features, the total asymptotic cost for the F1 computation is $O(m \cdot (n + n_c))$. As $n \geqslant n_c$ (there is at least one example per class), $O(m \cdot (n + n_c))$ can be reduced to $O(m \cdot n)$.

3.1.2 *Volume of the Overlapping Region (F2)*

Computes the overlap of the distribution of the features values within the classes. First, the maximum and minimum values of the feature from each class is calculated. The value is normalized by the total size of the interval. Then, the values obtained for

each feature is multiplied. The higher the F2 value, the greater the amount of overlap between the problem classes and the more complex is the problem. Note that if there is at least one non-overlapping feature, the numerator must be null. F2 value is given by Equation 4:

$$F2 = \prod_{i}^{m} \frac{\max(0, \min\max(f_i) - \max\min(f_i))}{\max\max(f_i) - \min\min(f_i)}, \tag{4}$$

where:

$$\min\max(f_i) = \min(\max(f_i^{c_1}), \max(f_i^{c_2})), \tag{5}$$
$$\max\min(f_i) = \max(\min(f_i^{c_1}), \min(f_i^{c_2})), \tag{6}$$
$$\max\max(f_i) = \max(\max(f_i^{c_1}), \max(f_i^{c_2})), \tag{7}$$
$$\min\min(f_i) = \min(\min(f_i^{c_1}), \min(f_i^{c_2})). \tag{8}$$

The values $\max(f_i^{c_j})$ and $\min(f_i^{c_j})$ are the maximum and minimum values of each feature in a class c_j, respectively. Figure 8a present an example of a feature for which $F2 = 0$, that is, where there is no overlapping region for this feature and 8b shows an example where there is an overlap.

(a) Non-overlapping classes.

(b) Overlapping classes.

Figure 8: Two possible scenarios for F2.

The F2 value can become very small depending on the number of operands. Therefore, it is highly dependent on the number of features a dataset has. For multiclass classification problems, a pairwise analysis of the classes (one versus one - OVO - decomposition) is performed and an average value is returned.

The asymptotic cost of this measure is $O(m \cdot n \cdot n_c)$, considering a OVO decomposition in the case of multiclass problems.

3.1.3 *Maximum Individual Feature Efficiency (F3)*

F3 measures how much a feature contributes to separate two classes. The individual efficiency is calculated by the quotient of the number of points that are out of the overlapping area for a given feature and the total number of instances. The maximum number among all features is returned, which corresponds to the most discriminative feature in the dataset.

$$F3 = \max_{i=1}^{m} \frac{\sum_{j=1}^{n} I(x_{ji} < \max\min(f_i) \vee x_{ji} > \min\max(f_i))}{n}, \tag{9}$$

where the numerator gives the number of examples that are **not** in the overlapping region for feature f_i. I is a indicator function, which returns 1 if its argument is true and 0 otherwise. $\max\min(f_i)$ and $\min\max(f_i)$ are the same of the F2 measure.

In this work, in order to standardize the results a mathematical manipulation was done. Our results were given according to

$$F3' = 1 - F3 \tag{10}$$

The lower the F3' value, the simpler is the problem, indicating few overlap between different classes. Again, for multiclass classification problems, a pairwise analysis of the classes is performed and an average value is returned.

As with F2, the asymptotic cost of the F3 measure is $O(m \cdot n \cdot n_c)$.

3.1.4 *Collective Feature Efficiency (F4)*

Collective Feature Efficiency assesses how features work together by applying F3 iteratively. The most discriminative feature according to F3 is selected, then all instances that can be separated by this feature are eliminated from the dataset and the procedure starts again , until all features have been considered or all examples have been removed. F4 returns the ratio of instances that have been discriminated.

Equation 11 denotes this procedure:

$$F4 = \frac{n_{\sim o}(f_i)_{T_i}}{n}, \tag{11}$$

where $n_{\sim o}(f_i)_{T_i}$ measures the number of points out of the overlapping region of feature f_i for the dataset T_i. At each round, f_i is the current most discriminative feature in T_i. T_i can be expressed as:

$$T_1 = T, \tag{12}$$

$$T_i = T_{i-1} - \{\mathbf{x}_j | x_{ji} < \max\min(f_{i-1}) \vee x_{ji} > \min\max(f_{i-1})\} \tag{13}$$

That is, T_i is continuously reduced by removing all examples that are already discriminated by the previous considered feature f_{i-1}. This is done until all features are considered or all examples are discarded.

F4 applies the F3 measure multiple times and at most it will iterate for all input features, resulting in a worst case asymptotic cost of $O(m^2 \cdot n \cdot n_c)$.

In this work, in order to standardize the results a mathematical manipulation was done. Our results were given according to

$$F4' = 1 - F4 \tag{14}$$

Lower F4' values reflect that more instances can be discriminated with fewer features, thus being simpler.

3.2 MEASURES OF SEPARABILITY OF CLASSES

These measures estimate the complexity of the decision border and the overlapping of the classes.

3.2.1 *Sum of the Error Distance by Linear Programming (L1)*

This measure aims to investigate whether the problem is linearly separable. The idea is that problems that can be separated by a hyperplane are simpler than problems that are not linearly separable. L1 is computed by the sum of the distance ε_i of instances that are incorrectly classified to the separating hyperplane, as indicated in Equation 15. A linear Support Vector Machine (SVM) can be used to obtain the hyperplane. Null L1 values indicate a linear separable problem, thus being simpler.

$$L1 = \frac{1}{n}\sum_{i=1}^{n}\varepsilon_i, \tag{15}$$

where the ε_i values are determined in the SVM optimization process and n is the total number of examples. It should be noticed that the ε_i values of correctly classified examples is null.

In multiclass classification problems decomposed according to OVO, this cost would be $O(n_c^2 \cdot (\frac{n}{n_c})^2)$, which resumes to $O(n^2)$ too.

3.2.2 *Error Rate of Linear Classifier (L2)*

L2 measure computes the error rate of a linear SVM classifier. Higher L2 values indicates a greater number of incorrect classifications, indicating that the problem is non linear, therefore more complex. Both L1 and L2 measures are defined for binary classification problems. In multiclas problems, some decomposition must be done a priori. L2 is given by the Equation 16:

$$L2 = \frac{\sum_{i=1}^{n} I(h(\mathbf{x}_i) \neq y_i)}{n} \tag{16}$$

where $h(\mathbf{x})$ is the L2 linear classifier, n is the total number of examples and y_i is the class of $\mathbf{x}_i$.

The asymptotic cost of L2 is the same of L1, that is, $O(n^2)$.

3.2.3 *Fraction of Borderline Points (N1)*

The fraction of borderline points measure (N1) assesses the complexity by regarding on the critical points of a dataset: those located in the decision boundary. In order to achieve that, a Minimum Spanning Tree (MST) is built from the dataset, where each vertex corresponds to one of the training instances and the edges correspond to the weighted distance between the connected instances. Afterwards, the percentage of

vertexes connected which belong to different classes is calculated. Finally, this value is divided by n, the total number of examples in the dataset, as given in Equation 17.

$$N1 = \frac{1}{n}\sum_{i=1}^{n} I((\mathbf{x}_i, \mathbf{x}_j) \in MST \wedge y_i \neq y_j) \tag{17}$$

Higher N1 values indicate a more complex decision boundary, indicating a more complex classification problem. An example is presented in Figure 9, where the different shapes/colors indicate different classes.

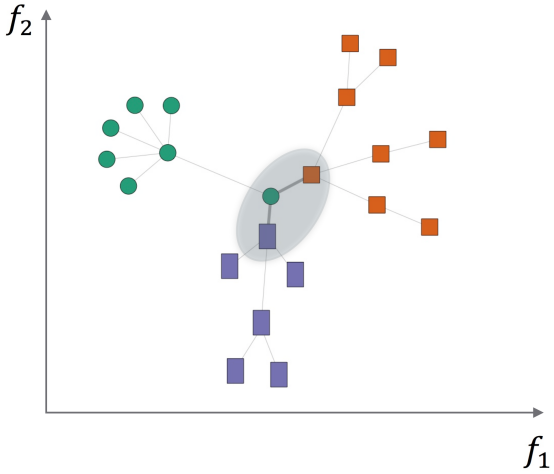


Figure 9: Minimum Spanning Tree showing the vertex connected to examples of different classes.

To build the graph from the data, it is necessary to first compute the distance matrix between all pairs of elements, which requires $O(m \cdot n^2)$ operations. Next, using *Prim's algorithm* for obtaining the MST requires $O(n^2)$ operations in the worst case. Therefore, the total asymptotic complexity of N1 is $O(m \cdot n^2)$.

3.2.4 *Ratio of Intra/Extra Class Nearest Neighbor Distance (N2)*

N2 measure assesses, for each training example, the ratio between the distance of the nearest neighbor ($k = 1$) from the same class and the nearest neighbor from a different class. By doing this, it reflects the distribution within classes and in the decision boundary. N2 is given by Equation 18:

$$N2 = \frac{\sum_{i=1}^{n} d(\mathbf{x_i}, NN_{y_i}(\mathbf{x}_i))}{\sum_{i=1}^{n} d(\mathbf{x_i}, NN_{y_j \neq y_i}(\mathbf{x}_i))}, \tag{18}$$

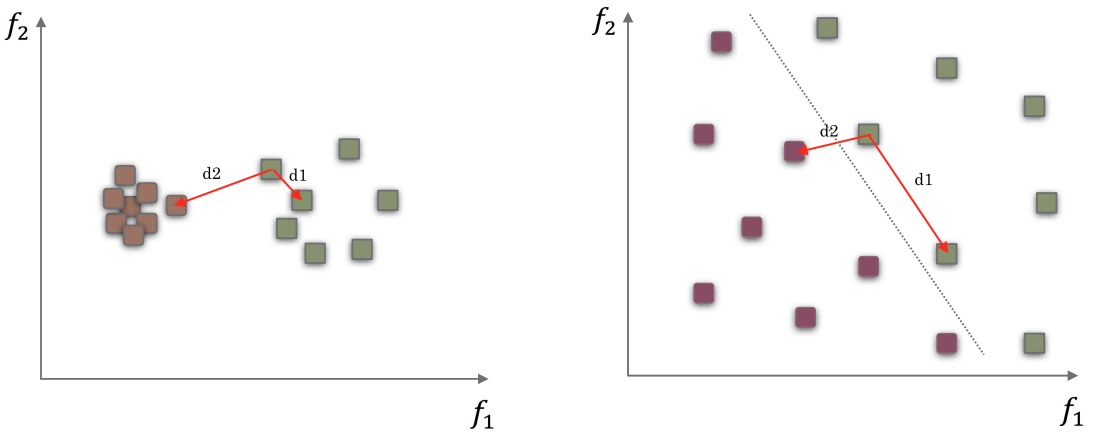
where $d(\mathbf{x_i}, NN_{y_i}(\mathbf{x}_i))$ corresponds to the distance of example $\mathbf{x}_i$ to its nearest neighbor from its own class y_i and $d(\mathbf{x_i}, NN_{y_j \neq y_i}(\mathbf{x}_i))$ represents the distance of $\mathbf{x}_i$ to the closest neighbor from another class $y_j \neq y_i$.

In this work, in order to standardize the results a mathematical manipulation was done. Our results were given according to

$$N2' = 1 - \left(\frac{1}{1+N2}\right) \tag{19}$$

Lower N2' values indicate a simpler problem, showing that instances from the same class are tight together rather than closer to instances from opposite classes.

An example of N2 computation for two examples in two learning datasets is presented in Figure 12.



(a) N2 example (b) N2 example

Figure 10: Distance between examples of the same and opposite class.

Computation of N2 requires obtaining the distance matrix between all pairs of elements in the dataset, which requires $O(m \cdot n^2)$ operations.

3.2.5 *Error Rate of the Nearest Neighbor Classifier (N3)*

N3 measure computes the error rate of an 1NN classifier using a leave-one-out strategy. It compares the prediction of the classifier with the real value of the analyzed instance. It sums up 1 if the prediction is incorrect and 0 otherwise. Afterwards, the total number of incorrect classified instances is divided by the total number of instances in the dataset. This procedure is denoted by Equation 20:

$$N3 = \frac{\sum_{i=1}^{n} I(NN(\mathbf{x}_i) \neq y_i)}{n}, \tag{20}$$

where $NN(\mathbf{x}_i)$ represents the nearest neighbor prediction for example $\mathbf{x}_i$. High N3 values represent that many examples are close to examples of other classes, making the problem more complex.

N3 requires $O(m \cdot n^2)$ operations.

3.3 MEASURES OF GEOMETRY, TOPOLOGY AND DENSITY

These measures capture the distribution of the examples within the classes and the density of the dataset.

3.3.1 *Non Linearity of Linear Classifier (L3)*

Based on Hoekstra and Duin [31] method, L3 first creates a new dataset by interpolating pairs of training examples of the same class. Herewith, two examples from the same class are chosen randomly and they are linearly interpolated (with random coefficients), producing a new example. Then a linear classifier is trained on the original data and has its error rate measured in the new data points. Higher values of this index indicate a greater complexity.

Letting $h_T(\mathbf{x})$ denote the linear classifier induced from the original training data T, the L3 measure can be expressed by:

$$L3 = \frac{1}{l}\sum_{i=1}^{l} I(h_T(\mathbf{x}'_i) \neq y'_i), \tag{21}$$

where l is the number of interpolated examples, $\mathbf{x}'_i$ are the interpolated examples and their corresponding labels are denoted by y'_i.

The asymptotic cost of this measure is dependent on both the induction of a linear SVM and the time taken to obtain the predictions for the l test examples, resulting in $O(n^2 + m \cdot l \cdot n_c)$.

3.3.2 *Non Linearity of the Nearest Neighbor Distance (N4)*

Similar to L3, but using the 1-NN classifier instead.

$$N4 = \frac{1}{l}\sum_{i=1}^{l} I(NN_T(\mathbf{x}'_i) \neq y'_i), \tag{22}$$

where l is the number of interpolated points. Higher N4 values are indicative of problems of greater complexity.

The asymptotic cost of computing N4 is $O(m \cdot n \cdot l)$ operations, as it is necessary to compute the distances between all possible testing and training examples.

3.3.3 *Fraction of Hypersphere Covering Data (T1)*

T1 builds hyperspheres centered at each one of the examples. Their radius is increased until they reach an example of another class. Smaller hyperspheres contained in larger hyperspheres are eliminated. T1 then returns the ratio between the number of these hyperspheres and the total number of examples in the dataset:

$$T1 = \frac{\sharp Hyperspheres(T)}{n} \tag{23}$$

where $\sharp Hyperspheres(T)$ gives the number of hyperspheres that can be formed in the dataset.

The idea is to obtain an adherence subset of maximum order for each example such that it includes only examples from the same class. Subsets included into other

ones are discarded. Each resulting subset corresponds to an hypersphere. Fewer hyperspheres are obtained for simpler datasets. This reflects the fact that data from the same class are densely distributed and near each other. Herewith, this measure also captures the distribution of data within the classes. The asymptotic cost of this measure can reach up to $O(m \cdot n^2)$.

3.4 RELATED WORK

The need for data complexity evaluation arose after an extended study on classification performance and the lack of understanding into the reasons why a given classifier performance is better in a dataset and worse in another. One of the first works to employ data complexity measures in FS was [51]. In this work some data complexity measures are defined, based on partitioning the feature space into multiple resolutions that vary in number of cells and volume. Concepts of purity, neighborhood separability, collective entropy, and data compactness are presented. The first two are related to data complexity, which is based on the difficulty for a classification algorithm in drawing a decision boundary. FS is performed by maximizing a neighborhood separability measure. In [20] another preliminary work on FS is also performed by using a definition of classifiability introduced in their work, which characterizes the relative easy with which some labeled data can be classified. In [32] a measure named neighborhood decision error rate is embed into a FS algorithm. All previous work defined new measures of classification complexity, which were then applied in FS.

In [3] the data complexity measures from Ho and Basu [30] are used to understand FS effects in magnetic resonance spectrum data classification. They find that a Genetic Algorithm feature subset selection technique made the classification problem easier, with less borderline examples and more concentrated and spherical classes. The work [46] proposes to quantify whether FS effectively changes the complexity of the original classification problem. For such, the authors monitor the values of some complexity measures in datasets with dimensionality reduced by two popular FS techniques. In both cases, they found that FS was able to increase class separability in the reduced subspaces. In [52] a similar study is performed, by monitoring changes in class separability due to the presence of irrelevant features in a dataset. In a controlled set of experiments, they observed that class separability has changed after the elimination of irrelevant features.

In [11], the authors propose a new method for distributed FS. Their method use some complexity measures (F1, F3, L1, N1 and N2) to compute a theoretical complexity of a feature which is further used in a merging procedure. The main advantage is that the new framework is independent of a classifier, and suitable for any FS algorithm. In [50] three complexity measures (F1, F2 and F3) are used to compute a threshold in feature ranking. The idea is to relieve the user from deciding for a fixed threshold. Experiments were run in six microarray datasets that imposed challenges to the technique. Afterwards, the same authors expanded their automated threshold to FS ensemble algorithms [2]. Using the same complexity measures, they compared the use of multiple FS algorithm in two different approaches: combining FS results before and after thresholding. In both cases, the complexity measures are used to

evaluate the feature subset generated and choosing the one that returns the simplest subset.

In this work we build on the previous work by employing the original data complexity measures of Ho and Basu [30] to guide some simple FS algorithms: (i) univariate, in which the complexity measures are employed to evaluate the importance of each feature independently and ranking them accordingly; and (ii) multivariate, in a forward and backward best-first FS. Results are then combined in a univariate-multivariate feature selection (UMFS) algorithm.

4

PROPOSAL AND EXPERIMENTS

This chapter describes the FS algorithms designed in this work, which rely on the complexity measures in order to evaluate the importance of the features. Basically, the complexity measures can be used in two kinds of setup:

- Univariate: in this case, the complexity measure is used in the evaluation of each feature individually. A score of the feature relevance is obtained, which can then be used to rank the input features. The feature overlapping measures were used in this setup, since their formulation evaluates each feature individually;
- Multivariate: two simple strategies were tested in this case, forward and backward selection. The values of a complexity measure are used to decide which feature must be added/removed at each step of these algorithms. Measures of separability of the classes were employed in this case;

Some previously work were done along this master's thesis and were published in [43], as a conference paper. These studies helped us to understand the behavior of all complexity measures in both univariate and multivariate setup. From there, we could observe that some complexity measures had a univariate nature and others a multivariate nature.

In order to combine multiple views regarding data complexity, we opted to implement a combined univariate-multivariate setup. Herewith, first the features are ranked by an univariate complexity measures. This can be done at a lower cost and allows removing irrelevant features that are contributing to increase data complexity. Next, the top ranked features are subject to a multivariate analysis using other complexity measure, which allows to take into account the joint relevance of the features.

Experiments were performed in order to identify the measures to be combined in this FS scheme. They are described next.

4.1 EXPERIMENT 1 - UNIVARIATE FS

Our premise in this work is that simpler problems should contain only relevant features and that the presence of irrelevant features will make the problem more complex. Hence, the identification and removal of irrelevant features from the dataset would make the classification problem simpler.

4.1.1 *Methodology*

In this experiment, we used the measures F1, F2, F3, and F4 to rank the input features for FS purposes. Afterwards, we evaluated the feature ranking results by three metrics. The first computes the precision of feature ranking when a threshold corresponding to the known number of relevant features is used to select the features. That is, it computes the percentage of features chosen that are indeed relevant among the

top-ranked features. The second evaluation metric computes the percentage of the ranking (Coverage %) that has to be regarded in order to retrieve all relevant features. That is, it divides the position of the worst ranked relevant feature by the total number of features in the dataset. The third metric is based on the AUC (Area Under the ROC curve) concept, which is independent of a particular threshold value on the number of chosen features. Given a ranking of features, the ROC curve is built considering the true positives (TP, the number of correctly identified relevant features) and false positives (FP, the number of irrelevant features incorrectly predicted as relevant) rates. Next, the area under the plotted curve is calculated. AUC values range from 0 to 1, where higher values indicate a better performance.

4.1.2 *Datasets*

The datasets used in the mentioned methodology are synthetic with relevant features known *a priori*. The reason for using synthetic datasets is two-fold: i) in controlled experiments it is possible to vary the experiment conditions in a systematic way, as adding noise and irrelevant features. This condition allows us to draw better conclusions and validate the advantages and disadvantages of an algorithm; ii) as mentioned, in this kind of dataset we already know which are the most relevant features, so we can assess the precision rate of a method by comparing its output with the known relevant features.

More information about these datasets are presented on Table 2. They were collected from [9]. Each dataset imposes a different challenge for the measures and can be organized into five broad groups, according to [9]:

- **Dealing with correlation** - CorrAL-500 contains four relevant features and an irrelevant feature that is very informative for classification. Most of the FS techniques tend to select this feature (number 5), since it is highly correlated with the class (75% of correlation), but yet it is not discriminant enough to distinguish the classes.
- **Dealing with noise in the input** - Led500no and Led500n15 allows testing robustness against noise in the input features. We are comparing the difference in performance between a version with no noise (Led500no) versus a version with 15% of noise (Led500n15).
- **Dealing with noise in the target** - Monk1, Monk2 and Monk3 datasets present 5% of misclassifications (label noise). This fact causes the classification accuracy to be higher when we do not select all relevant features.
- **Dealing with non-linearity** - XOR-500 is a well known non-linear dataset. Parity3+3 has 3 relevant and 3 redundant features.
- **Dealing with small ratio examples/features** - SD1, SD2 and SD3 datasets have high dimensionality with only a few relevant features. This scheme imposes a large difficulty: to find a few relevant features in a pool of irrelevant features.

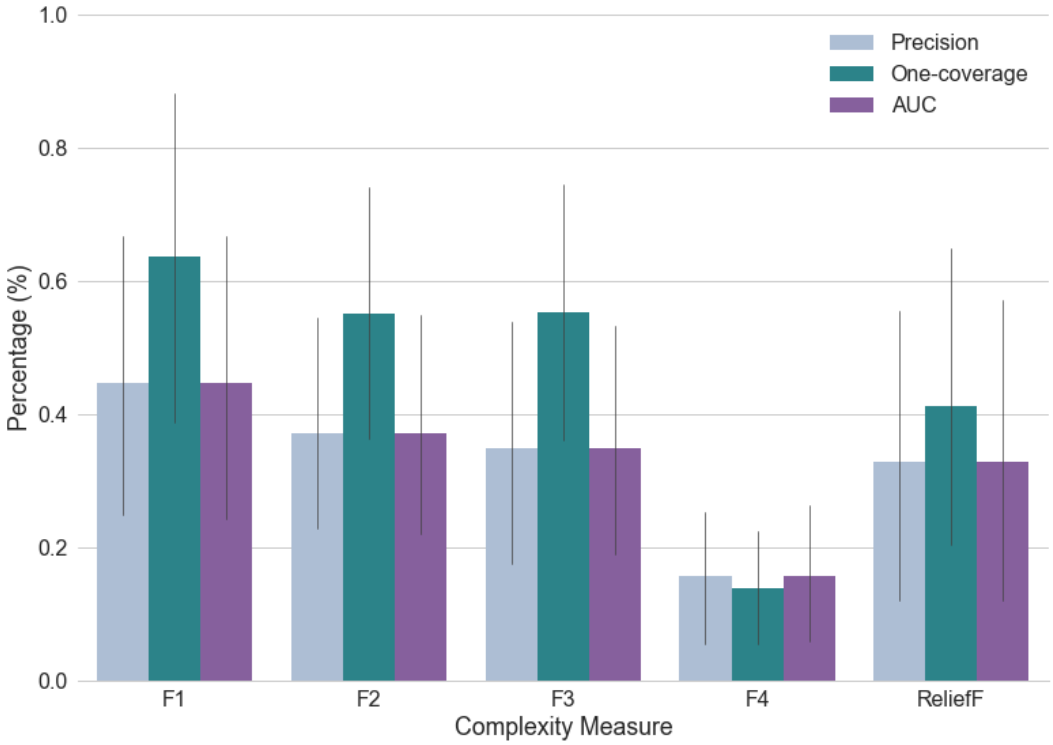
Table 2: Summary of the synthetic datasets.

Dataset	# Examples	# Features	Relevant Features	Noise	# Classes	% Majority Class
CorrAL-500	1000	500	1-4	0%	2	56.00%
Led500no	1000	500	1-7	0%	10	10.00%
Led500n15	1000	500	1-7	15%	10	11.00%
Monk1	500	100	1,2,5	0%	2	50.20%
Monk2	500	100	1-6	0%	2	66.40%
Monk3	500	100	2,4,5	5%	2	56.20%
Parity3+3	1000	500	1-3	0%	2	50.20%
SD1	300	4000	1-2	0%	3	33.60%
SD2	300	4000	1-4	0%	3	33.60%
SD3	300	4000	1-6	0%	3	33.60%
XOR-500	1000	500	1-2	0%	2	50.20%

4.1.3 *FS results*

The average results in terms of precision, 1 - coverage% (the complement of coverage % is taken to standardize the results) and AUC achieved by using the complexity measures F1, F2, F3, and F4 in feature ranking are shown in Figure 11. We also include the ReliefF [35] feature ranking technique as a baseline which is considered

Figure 11: Precision, 1 - Coverage % and AUC results in univariate FS for the synthetic datasets



the best option when there is no prior knowledge about the data [12]. ReliefF is a strong baseline already largely employed in FS. The complete results per dataset for all performance metrics and complexity measures are shown in Appendix A.1.

For all evaluation metrics plotted, the higher the values, the better the results. It is possible to notice that F1 achieved the best results for all metrics. F2 and F3 performed similarly, F2 had a better precision whereas F3 had a better coverage result. ReliefF was the forth in terms of FS performance. F4 performed poorly as expected by fig **??**. It was not able to reflect complexity properly and struggled to find relevant features.

Therefore, the F1 measure, which regards feature discrimination ability, was the most successful in retrieving the relevant features in the datasets investigated considering an univariate FS scheme. In fact, this measure is already largely employed in FS [33].

F1 measure was able to correctly select the relevant features in two datasets: Led500no and Led500n15. These are variants of a dataset with a gradual addition of noise (0% in the former and 15% on the latter). F2 and F3 were also able to select the relevant features in Led500no, but were not able to select the relevant features in Led500n15. This show a robustness against noise in F1 when compared to F2 and F3. ReliefF could also select the relevant features in Led500no and achieved a better precision result in Led500n15 compared to F2 and F3. Surprisingly, F4 was able to select the relevant features in XOR-500.

Even though any complexity measure could not correctly select all the relevant features for datasets SD1, SD2 and SD3, they all had high Coverage. Meaning that the relevant features were in top-positions. F1 was able to identify 0%, 50% and 50% of the relevant features in SD1, SD2 and SD3 datasets, respectively. Whereas F3 and F4 were able to identify 33%, 0% and 50% on the same datasets, respectively.

F1 was the only measure that could find at least one relevant feature in Monk1, Monk2 and Monk3 datasets. F2, F3 and F4 had the same precision on these datasets, 0.67%, 0.33% and 0%, respectively. But F1 performed poorly on dataset Parity3+3. In fact, F1 measures the discrimination power relative to each of the feature axis, while in this dataset a combination of the features should be regarded instead.

Concluding, F1 was the best measure for univariate FS, being robust against noisy data and effective even for very large datasets. F2 and F3 presented some sensitivity to noise, but they performed well for some challenging datasets such as SD1, SD2 and SD3. F4 achieved the lowest FS power among the complexity measures as expected from initial experiments.

4.2 Experiment 2 - Multivariate FS

In this FS setting, we evaluate the complexity measures regarding their ability to select the relevant features in a multivariate setup, using the outputs from the univariate setup. The idea is to evaluate which measure will perform better in combination with F1, based on the initial results presented previously. For selecting the initial threshold employed in the univariate ranking, we used a method described in [22]. The number of features selected corresponds to the one that accumulates 95% of the total feature importance value, as measured by F1.

For multivariate FS, two approaches are tested and compared: forward and backward selection. The idea is to identify whether forward or backward selection can

return better values of the performance metrics. Herein, we run the experiment over the results of the univariate FS and part of the relevant features can be already discarded. Therefore, we do not have the entire list of features to determine precisely the performance metrics. It is worth noting that we cannot expect 100% precision or coverage or AUC score for some datasets. This is because an univariate FS based on F1 was already performed and a subset of features was selected. We cannot guarantee that all relevant features are among those selected and, therefore, it will be not able to improve the results. Take for instance CorrAL-500 dataset. If univariate FS selected the features 0, 1, 2, 200, 300, 400 and 500, in this multivariate FS, the best precision that can be achieved is 0.42857.

4.2.1 *Methodology*

Due to the univariate nature of F1, F2, F3 and F4 measures, they are not included in these experiments. In fact, they would always return the same complexity value in all steps of forward or backward selection, since they evaluate each feature individually and returns the maximum of the discriminant values found. For instance, taking F1 and forward selection, the first step would choose the most discriminative feature and the addition of anyone of the other features in further steps would maintain the same F1 value.

Complexity measures L1, L2, N3, L3, L4 and N4 were also not included in these experiments because they all use a classifier in some part of the algorithm. L1 and L2 use SVM to compute complexity. Both are based on classifications errors, the former calculates the distances of instances that are incorrectly classified and latter measures the error rate of the classifier. N3 does the same as L2 but using the 1-NN classifier. L3 creates a new datasets by interpolating pairs of points from the original dataset and then uses SVM to compute the error rate on this new data. N4 does the same as L3 but using the 1-NN classifier instead. In addition, T1 creates hyperspheres centered on each example and eliminates smaller hyperspheres that are contained in larger ones. This shall reflect whether the data from the same class are densely distributed and near each other. However, it did not present good results and showed a high runtime. Therefore, we use only N1 and N2 complexity measures in this setup, which are independent of a specific classifier and are faster to compute.

Algorithm 1: Forward FS.

Require: Dataset T, complexity measure C and threshold in number of features $n\_feat\_to\_keep$;

$features\_in = \emptyset$;
while $|features\_in| < n\_feat\_to\_keep$ **do**
  $features\_out = all\_features - features\_in$;
  for $f \in features\_out$ **do**
    $f_t = features\_in + f$;
    Compute complexity measure $C(T[f_t])$
  end for
  $features\_in = features\_in \cup f$ with the best complexity measure value
end while

Algorithm 2: Backward FS.

Require: Dataset T, complexity measure C and threshold in number of features n_feat_to_keep;

1: features_in = all_features;
2: **while** |features_in| > n_feat_to_keep **do**
3: **for** f ∈ features_in **do**
4: f_t = features_in − f;
5: Compute Complexity Measure C(T[f_t])
6: **end for**
7: features_in = features_in − f with the worst complexity measure value
8: **end while**

A pseudo-code of the forward algorithm implemented is presented in Algorithm 1, whilst the backward FS scheme is presented in Algorithm 2. In both algorithms, features_in will be returned as the set of selected features. In forward selection, this set is initially empty (line 1 of Algorithm 1) and is increased until its size reaches a given threshold (n_feat_to_keep). At each step, the feature to be included in features_in is the one which leads to the best complexity measure value when combined to the features already in features_in. Therefore, only features which contribute more in reducing the problem complexity according to the complexity measure computation are selected at each step. In backward selection, features_in contains all features initially and this set is continuously decreased until the threshold on the number of features is reached. The feature to be removed at each step is the one which simplifies the problem more when it is disregarded. We denote this as the feature with the worst complexity measure value in line 7 of Algorithm 2.

Tables 3 and 4 illustrate the operation of the forward and backward FS algorithms, respectively, for a reduced version of CorrAL-500 dataset (CorrAL-500 with only 6 features: 4 relevant, 1 highly correlated and 1 irrelevant). In both executions the N1 measure was employed as a feature importance measure. Each table shows the set features_in under evaluation and the N1 values at each step of the algorithm. Both algorithms are stopped when four features are selected, which is the known number of relevant features in CorrAL dataset. In Table 3, for instance, at the first step feature 4 will be chosen first, since the N1 value using this feature is the lowest (recall that lower N1 values indicate the problem is simpler). Afterwards, each of the other features left out are evaluated jointly to feature 4. Again, the feature with the lowest measure value is chosen. The algorithm proceeds with features 4 and 5 and seek for the best combination with each of the remaining features. The final solution is the combination [4 5 0 1]. It contains two relevant features, one highly correlated with the class label and one irrelevant feature. In Table 4, the removal of feature 4 from the initial set leads to the best N1 value. In the second step, feature 0 is removed. At the end, the set [1 2 3 5] is chosen. It contains three relevant features and the highly correlated feature. Given that CorrAL dataset has 6 features and 4 relevant features, it is faster to compute by backward selection compared to forward selection. In this specific case, it is faster and more accurate too.

Table 3: CorrAL dataset forward selection using the N1 Measure

Features_in	N1	Status
[]	-	Initial solution
[0]	0.53125	
[1]	0.68750	
[2]	0.59375	
[3]	0.65625	
[4]	0.50000	Feature added
[5]	0.75000	
[4 0]	0.53125	
[4 1]	0.53125	
[4 2]	0.53125	
[4 3]	0.53125	
[4 5]	0.43750	Feature added
[4 5 0]	0.37500	Feature added
[4 5 1]	0.56250	
[4 5 2]	0.65625	
[4 5 3]	0.56250	
[4 5 0 1]	0.46875	Feature added
[4 5 0 2]	0.53125	
[4 5 0 3]	0.53125	
[4 5 0 1]	0.46875	Final solution

4.2.2 *FS results*

The average results for all performance metrics achieved by the complexity measures N1 and N2 in forward and backward FS are shown in Figures 12a and 12b, respectively. The complete results per dataset for all performance metrics and complexity measures for backward and forward selection are shown in Appendix A.2 and A.3, respectively. ReliefF [35] was added again as a baseline, for both forward and backward selection. For all FS performance metrics, the higher the value, the better. It is possible to notice that N2 in forward selection returns better results than N1 in both directions. N1 returns very similar results in either forward or backward selection. As we pointed out previously, starting with a full set has the advantage of using all information provided to search for relevant features. However, our results show that the amount of irrelevant features contained in the datasets analyzed actually hinder the process of the identification of the relevant features. In fact, in most of the datasets the number of irrelevant features is too large compared to the number of relevant features.

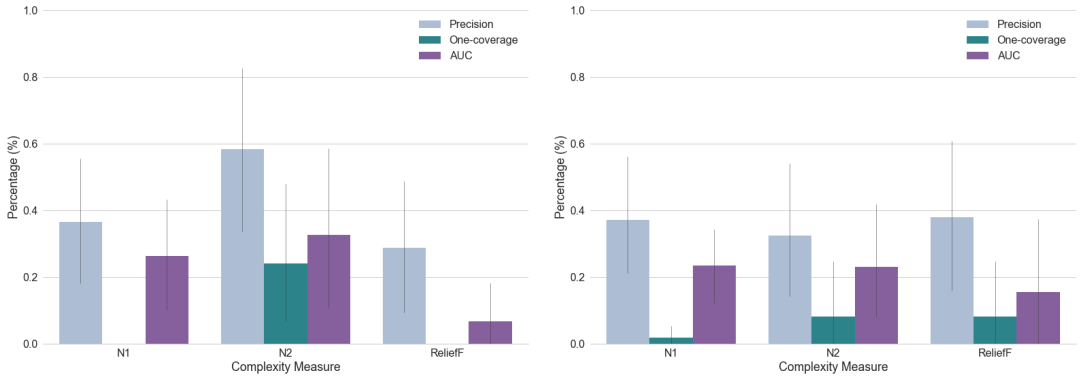
Within the backward selection, N1 was the most successful complexity measure, being able to correctly select the relevant features in CorrAL-500. N1 was also the

Table 4: CorrAL dataset backward selection using the N1 measure

Features_in	N1	Status
[0 1 2 3 4 5]	-	Initial Solution
[1 2 3 4 5]	0.50000	
[0 2 3 4 5]	0.43750	
[0 1 3 4 5]	0.50000	
[0 1 2 4 5]	0.53125	
[0 1 2 3 5]	0.34375	Feature removed
[0 1 2 3 4]	0.50000	
[1 2 3 5]	0.43750	Feature removed
[0 2 3 5]	0.62500	
[0 1 3 5]	0.59375	
[0 1 2 5]	0.53125	
[0 1 2 3]	0.50000	
[1 2 3 5]	0.43750	Final solution

complexity measure that found relevancy in most of the datasets. It did not find any relevant feature in only three datasets (Monk2, Parity3+3 and XOR-500), while N2 did not find any relevant feature in four datasets (Monk2, Parity3+3, SD3 and XOR-500) and ReliefF failed in five datasets (Monk2, Parity3+3, SD2, SD3 and XOR-500). On the other hand, both N2 and ReliefF were able to identify all the relevant features in SD1, in which N1 could only identify one single relevant feature (among two). This fact shows that N2 has some degree of robustness against high-dimensionality when compared to N1.

N1 and N2 presented very similar AUC scores. Both achieved the same results for most of the datasets, except SD1 and SD3, where N1 was better in the first and N2 in the second. In respect to non-linear datasets, all complexity measures suffered in



(a) Forward Selection

(b) Backward Selection

Figure 12: Precision, 1 - Coverage % and AUC results in multivariate FS for synthetic datasets.

detecting the relevant features. On Led datasets, where noise is added, we could not see a significant loss of performance. This shows that both complexity measures are quite robust to noise.

Within the forward selection, N2 was the most successful measure, being able to correctly select the relevant features in four datasets (CorrAL-500, SD1, SD2 and SD3). It showed robustness to high dimensionality, since it could select relevant features in SD's datasets where relevant features are only 0.075% of all features included. In this setup, no other measure (N1 and ReliefF) could successfully select all relevant features of any dataset. In fact, N1 could not find any relevant feature in four datasets (Monk2, Parity3+3, SD1 and XOR-500) and ReliefF in six (Monk2, Parity3+3, SD1, SD2, SD3 and XOR-500).

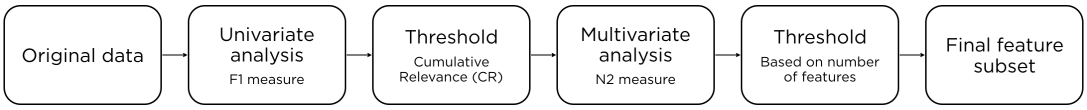
In Led500no and Led500n15, all measures scored the same precision values. This fact indicates that N1, N2 and ReliefF have some degree of robustness against noise. Among these, N2 and ReliefF had the same precision value of 0.7143 and N1 achieved a precision of 0.4286. Regarding Monk1, Monk2 and Monk3, which have noise on the target, all of the three feature importance measures had the same precision score of 0.3333, 0.0000 and 0.6667, respectively. In Parity3+3 and XOR-500, all measures failed in finding relevancy when combined to F1.

Herewith, we can conclude that forward selection perform better than backward selection. The main reason is that although the backward selection contains all the available information, this ended up hindering feature reelvance. Among the complexity measures tested, N2 outperformed the others in all scenarios drawn.

4.3 EXPERIMENT 3 - COMBINED FEATURE SELECTION

In this experiment, we combine both univariate and multivariate into a FS algorithm (UMFS). Univariate analysis will be performed based on F1 complexity measure to pre-select top ranked features. The threshold defined is based on [22]. Afterwards, a forward multivariate analysis will be performed using the N2 complexity measure with a stopping criteria based on [10], described in Section 2.1.5. Our UMFS proposal is shown in Figure 13. Herein, we add classical and microarray datasets too. In order to evaluate the effectiveness of FS in this section, the classification performance achieved with the selected subset of features is combined to that achieved using all features. The Support Vector Machines (SVM) and Random Forests (RF) classification techniques are used in this evaluation with standard scikit-learn [45] package parameters (SVM: $C = 1.0$, $kernel = rbf$, $degree = 3$, $gamma = 1/n\_features$ and RF: $n\_estimators = 10$, $criterion = gini$, $max\_depth = None$, $min\_samples\_split = 2$, $max\_features = auto$)

Figure 13: Univariate-multivariate feature selection (UMFS) algorithm framework.

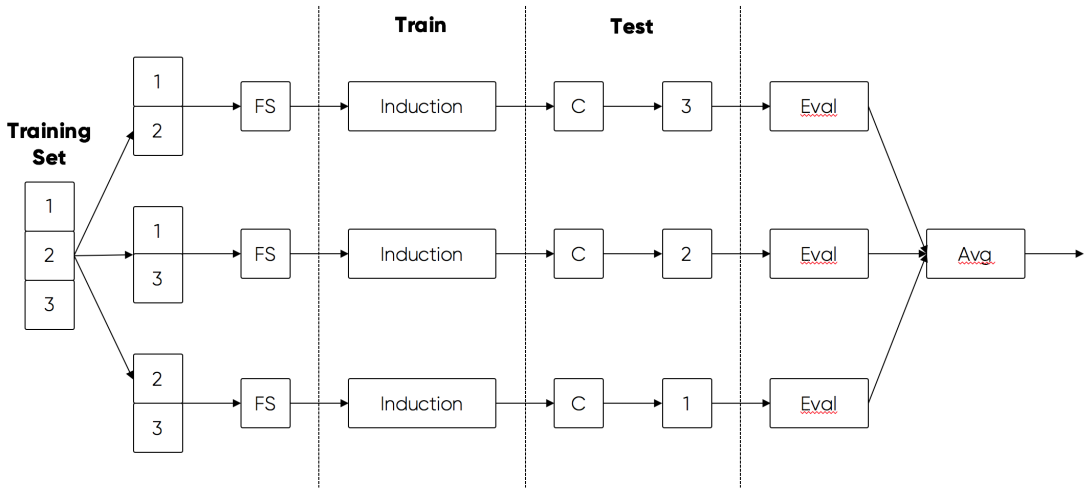


4.3.1 *Methodology*

We build a new FS algorithm (UMFS) based upon our previous experiments. As one of the reasons to use FS is to increase the classification performance, we compare accuracy before and after FS in order to certify whether it could improve or at least maintain the same predictive performance while using a reduced number of features. In high-dimensional domains, SVM is one of the most used models [49] due to its robustness. For comparison purposes, we also added the Random Forest classifier. Experimental evidence has shown that decision trees, such as C4.5, exhibit a degradation in the performance when faced with many irrelevant features [9]. Since the objective is to evaluate FS and not to achieve the best accuracy, no refinement on the classifiers was done. We used standard parameter values from the scikit-learn package [45].

We applied stratified k-fold cross validation in order to assess classification performance with FS. In this case, we implemented our own algorithm in order to apply FS in every fold. The idea is to guarantee an average result due to possible differences in each fold that can lead to a different feature subset. Figure 14 shows the stratified k-fold cross validation approach employed. FS denotes the feature selection algorithm, which is applied on the training sets. C denotes the classification model built, which is evaluated (Eval) in the corresponding test set. An average (Avg) performance value is then computed.

Figure 14: K-fold cross validation method for classification with feature selection (k=3 in this example) [36].



In addition to accuracy, we will evaluate the classification results with Cohen's Kappa Coefficient (κ) [15]. Kappa is a statistical measure computed as:

$$\kappa = \frac{P_o - P_c}{1 - P_c} \tag{24}$$

where P_o is the proportion of observed agreements between the predictions and the actual expected labels and P_c is the proportion of agreements expected by chance. This measure may complement the accuracy values, mainly for imbalanced datasets.

4.3.2 *Datasets*

In this experiment we also added more datasets so we can evaluate scenarios where we do not know the relevant features *a priori*. Three datasets are from the UCI Repository [17] and contain a higher number of samples when compared to the number of features (Table 5). They are more similar to situations we face on day-to-day work with features ranging from 42 to 501, and samples ranging from 2,400 to 67,557. They also contain problems like class imbalance [28] and nonlinearity. Other seven datasets added are from DNA microarray, in which the number of features is much higher that the number of samples (see Table 6). These datasets are a challenge to feature selection because they present a large number of irrelevant features and a very few number of examples. A common drawback in this kind of datasets is that we usually have more data about healthy patients causing class imbalance, like in the Lung dataset where the majority class has 82.87% of the examples. The class-imbalance problem may affect the classification techniques negatively, since they tend to build models focused on the majority class only [28]. All datasets used have numerical attributes only.

Table 5: Summary of the classical datasets.

Base	# Examples	# Features	# Classes	% Majority Class
Spambase [17]	4601	58	2	60.60%
Connect4 [17]	67557	42	3	65.83%
Madelon [17]	2400	501	2	50.12%

Table 6: Summary of the microarray datasets.

Base	# Examples	# Features	# Classes	% Majority Class
11 Tumors	174	12534	11	15.52%
Colon	62	2001	2	64.52%
Dlbcl	47	4027	2	51.06%
Leukemia	72	7130	2	65.28%
Leukemia 2	72	11226	3	38.89%
Lung	181	12534	2	82.87%
Ovarian	253	15155	2	64.03%

4.3.3 *Threshold*

As described in Section 2.1.5, two approaches were used in this experiment: one for the univariate based in on cumulative relevance [22]

$$CR_i = \sum_{i=1}^{m} c_i \tag{25}$$

where m is the total number of features. We then return the number of features that add up to γ percent of CR, where γ is some confidence level (e. g. $\gamma = 0.95$).

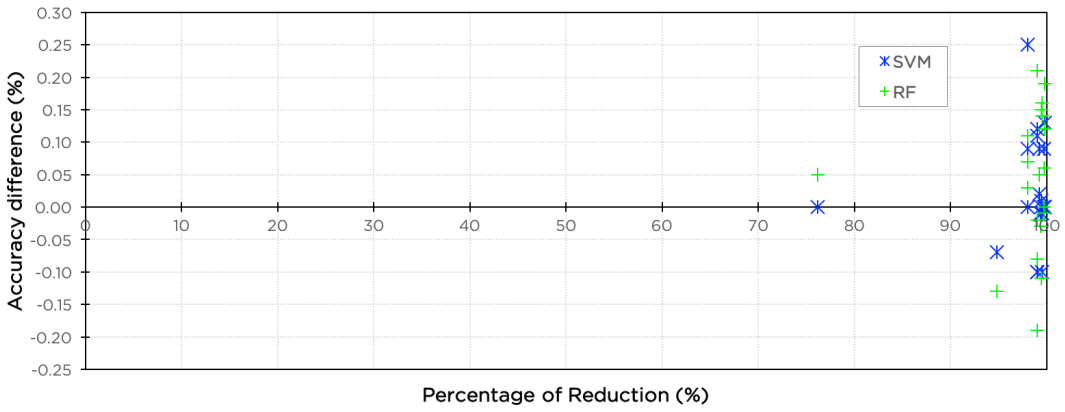
For the multivariate setup with a threshold based on the number of features m, as in [10].

- if $m < 10$, select 75% of the features;
- if $10 \leqslant m < 75$, select 40% of the features;
- if $75 \leqslant m < 100$, select 10% of the features;
- if $m \geqslant 100$, select 3% of the features.

4.3.4 *Combined FS results*

Figure 15 presents a compilation of all the results of the combined UMFS technique and the SVM and RF classifiers. It shows the percentage of reduction in the number of features of the dataset after FS (x axis) and the difference in accuracy after minus before FS was applied (y axis). Therefore, points above the zero in x line indicate a performance gain compared to using all features, while points bellow this line indicate a performance decrease. It is possible to see the significant reduction on the number of features selected in all cases. This is expected, due to the thresholds applied in FS, as described previously. In addition, is possible to see that the majority of the accuracy results were maintained or improved when compared to using all features. These results indicate that our proposed UMFS technique could successfully select the most relevant features for maintaining or improving classification performance.

Figure 15: Percentage of reduction versus the difference in accuracy taken from after and before FS.



The results obtained with the SVM classifier for all datasets are presented in Table 7. It shows a comparison of accuracy and kappa score (both with standard deviation) before and after FS. It also shows the number of features retained after FS (column # Feat (%)) and the percentage of reduction within parenthesis. Best results are highlighted on bold. In synthetic datasets, our UMFS algorithm could improve accuracy performance in eight out of 11 datasets. In these cases, we could also see an improvement on Cohen's kappa score, showing that the removal of irrelevant features

indeed reduced uncertainty and made the problem simpler. However, in the Monk2 dataset there was no improvement on classification performance nor on the kappa score. In SD2 and XOR-500, the increase in classification performance was slightly better. UMFS was not able to improve accuracy in Led500no, SD1 and SD3 datasets. For Led500no, a possible reason is that the final number of selected features was five, while the number of known relevant features is seven. Therefore, there is a loss of information that may be crucial for accuracy. For SD1 and SD3, it was expected that the algorithm would struggle to find the relevant features (2 and 6, respectively) between so much irrelevant information. Even though, the degradation of predictive performance in SD1 was minor.

In classical datasets, we could only maintain accuracy for Connect4 dataset. For the other two, there were a significant loss in accuracy. A possible reason is that FS selected only 5.17% and 0.99% of features for Spambase and Madelon, respectively. This is a great reduction and may have a cost of crucial information for classification, deteriorating the predictive results.

Table 7: Classification results for SVM before and after feature selection (best results in bold face).

	Original				Feature Selection				
Base	Accuracy (std)		Kappa (std)		# Feat (%)	Accuracy (std)		Kappa (std)	
CorrAL-500	0.80	(0.03)	0.58	(0.06)	5 (1.0)	**0.91**	**(0.02)**	**0.82**	**(0.03)**
Led500no	**0.63**	**(0.06)**	**0.58**	**(0.06)**	5 (1.0)	0.53	(0.02)	0.47	(0.02)
Led500n15	0.28	(0.03)	0.19	(0.03)	5 (1.0)	**0.40**	**(0.03)**	**0.33**	**(0.03)**
Monk1	0.65	(0.08)	0.30	(0.16)	2 (2.0)	**0.74**	**(0.05)**	**0.49**	**(0.09)**
Monk2	0.66	(0.01)	0.00	(0.01)	2 (2.0)	**0.66**	**(0.01)**	**0.00**	**(0.00)**
Monk3	0.67	(0.04)	0.30	(0.09)	2 (2.0)	**0.92**	**(0.03)**	**0.83**	**(0.06)**
Parity3+3	0.47	(0.05)	-0.07	(0.10)	4 (0.8)	**0.56**	**(0.05)**	**0.12**	**(0.10)**
SD1	**0.61**	**(0.07)**	**0.42**	**(0.10)**	21 (0.52)	0.60	(0.07)	0.40	(0.10)
SD2	0.61	(0.06)	0.42	(0.09)	22 (0.55)	**0.62**	**(0.08)**	**0.43**	**(0.12)**
SD3	**0.66**	**(0.09)**	**0.49**	**(0.13)**	23 (0.52)	0.56	(0.10)	0.34	(0.15)
XOR-500	0.53	(0.04)	0.07	(0.09)	4 (0.8)	**0.55**	**(0.04)**	**0.10**	**(0.07)**
Spambase	**0.73**	**(0.00)**	**0.32**	**(0.01)**	3 (5.17)	0.66	(0.00)	0.00	(0.00)
Connect4	0.50	(0.00)	0.00	(0.00)	10 (23.8)	**0.50**	**(0.00)**	**0.00**	**(0.00)**
Madelon	**0.84**	**(0.03)**	**0.67**	**(0.05)**	5 (0.99)	0.74	(0.02)	0.52	(0.04)
11 Tumors	**0.16**	**(0.03)**	**0.01**	**(0.02)**	79 (0.63)	0.15	(0.02)	0.01	(0.01)
Colon	0.65	(0.04)	0.00	(0.00)	4 (0.19)	**0.65**	**(0.04)**	**0.00**	**(0.00)**
Dlbcl	0.81	(0.18)	0.61	(0.36)	10 (0.24)	**0.94**	**(0.13)**	**0.88**	**(0.24)**
Leukemia	0.65	(0.06)	0.00	(0.00)	21 (0.29)	**0.65**	**(0.06)**	**0.00**	**(0.00)**
Leukemia 2	0.39	(0.06)	0.00	(0.00)	63 (0.56)	**0.39**	**(0.06)**	**0.00**	**(0.00)**
Lung	0.83	(0.01)	0.00	(0.00)	39 (0.31)	**0.83**	**(0.01)**	**0.00**	**(0.00)**
Ovarian	0.87	(0.05)	0.68	(0.14)	40 (0.26)	**0.96**	**(0.02)**	**0.92**	**(0.05)**

In microarray datasets, our approach was able to maintain or improve classification performance in six out of seven datasets. The only dataset it was not able to improve the results was "11 Tumors", in which the accuracy was worse by 0.01%. In fact, the standard deviation for accuracy and kappa score after FS was lower than on the original dataset. The lower the standard deviation, the more consistent the results. Considering a 99.37% size reduction, the results were quite good and could even be considered a tie. For Colon, Leukemia, Leukemia 2 and Lung datasets, our algorithm was able to maintain classification performance by selecting only 0.19%, 0.29%, 0.56% and 0.31% of the original features, respectively. In Dlbcl and Ovarian datasets, we were able to see a significant improvement on results, for both accuracy and kappa score (also with a lower standard deviation).

Table 8 shows the classification results for the Random Forest classifier before and after FS. Our FS method could improve the predictive power after FS in nine out of 11 synthetic datasets. The two datasets for which improvements were not achieved were Led500no and Led500n15, where the number of features selected was inferior to the number of relevant features contained in these datasets. The removal of the relevant features may have caused a loss of information required for an accurate classification. In general, there was an increase of 14% in accuracy after FS on the synthetic datasets.

In classical datasets, our method could only improve performance in one dataset: Connect4. There was a 5% improvement in accuracy and a reduction on standard deviation, showing a more consistent result. The initial accuracy in Madelon was already very high (94%), which we could not maintain after removing 99.01% of the features. The same happened to Spambase, after a removal of 94.83% of the features.

In microarray datasets, results were better in four out of seven datasets. Improvements were seen in Colon (+12%), Dlbcl (+19%), Leukemia (+6%) and Lung (maintaining accuracy with less features) datasets. The first three could improve accuracy with less than 0.3% of the original features. Although accuracy did not increase in the Lung dataset, it is possible to see an increase of 0.02 on Kappa Score and a reduction of standard deviation for both metrics, indicating more consistent results. On the other hand, a degradation of predictive accuracy was achieved in 11 Tumors (-3%), Leukemia 2 (-11%) and Ovarian (-1%) datasets.

Interestingly, the classification performance of the RF classifier was better than that of the SVM, which seems to have benefited more from FS.

Table 8: Classification results for Random Forest before and after feature selection (best result in bold face).

	Original				Feature Selection				
Base	Accuracy (std)		Kappa (std)		# Feat (%)	Accuracy (std)		Kappa (std)	
CorrAL-500	0.69	(0.03)	0.35	(0.07)	5 (1.0)	**0.90**	**(0.02)**	**0.80**	**(0.04)**
Led500no	**0.59**	**(0.06)**	**0.54**	**(0.07)**	5 (1.0)	0.51	(0.03)	0.45	(0.03)
Led500n15	**0.39**	**(0.07)**	**0.33**	**(0.07)**	5 (1.0)	0.37	(0.04)	0.30	(0.04)
Monk1	0.67	(0.04)	0.35	(0.07)	2 (2.0)	**0.74**	**(0.06)**	**0.49**	**(0.11)**
Monk2	0.61	(0.04)	-0.04	(0.11)	2 (2.0)	**0.64**	**(0.02)**	**0.02**	**(0.07)**
Monk3	0.81	(0.06)	0.63	(0.13)	2 (2.0)	**0.92**	**(0.03)**	**0.84**	**(0.05)**
Parity3+3	0.51	(0.04)	0.01	(0.08)	4 (0.8)	**0.56**	**(0.06)**	**0.13**	**(0.11)**
SD1	0.44	(0.06)	0.16	(0.09)	21 (0.52)	**0.58**	**(0.09)**	**0.36**	**(0.14)**
SD2	0.44	(0.08)	0.16	(0.12)	22 (0.55)	**0.59**	**(0.10)**	**0.38**	**(0.15)**
SD3	0.45	(0.09)	0.18	(0.13)	23 (0.52)	**0.61**	**(0.07)**	**0.42**	**(0.11)**
XOR-500	0.52	(0.06)	0.03	(0.13)	4 (0.8)	**0.57**	**(0.05)**	**0.15**	**(0.11)**
Spambase	**0.79**	**(0.01)**	**0.55**	**(0.01)**	3 (5.17)	0.66	(0.00)	0.00	(0.00)
Connect4	0.71	(0.05)	0.43	(0.11)	10 (23.80)	**0.76**	**(0.03)**	**0.52**	**(0.07)**
Madelon	**0.94**	**(0.01)**	**0.88**	**(0.03)**	5 (0.99)	0.75	(0.02)	0.53	(0.03)
11 Tumors	**0.74**	**(0.04)**	**0.70**	**(0.05)**	79 (0.63)	0.71	(0.07)	0.67	(0.08)
Colon	0.68	(0.15)	0.22	(0.35)	4 (0.19)	**0.80**	**(0.13)**	**0.58**	**(0.24)**
Dlbcl	0.73	(0.20)	0.44	(0.40)	10 (0.24)	**0.92**	**(0.16)**	**0.83**	**(0.35)**
Leukemia	0.90	(0.11)	0.71	(0.33)	21 (0.29)	**0.96**	**(0.06)**	**0.90**	**(0.15)**
Leukemia 2	**0.83**	**(0.08)**	**0.74**	**(0.13)**	63 (0.56)	0.72	(0.14)	0.57	(0.22)
Lung	0.97	(0.04)	0.88	(0.17)	39 (0.31)	**0.97**	**(0.03)**	**0.90**	**(0.10)**
Ovarian	**0.98**	**(0.03)**	**0.96**	**(0.06)**	40 (0.26)	0.97	(0.03)	0.94	(0.06)

5 CONCLUSION

In this chapter we present the main contributions and limitations of this work. Future possible works are also discussed.

5.1 CONTRIBUTIONS

In this work, we analyzed the use of data complexity measures as feature importance measures used to guide feature selection. For such, we defined a novel framework for FS combining univariate and multivariate statistics. The former is based on the F1 complexity measure, the maximum Fisher's Discriminant Ratio, that ranks features according to their discriminative power. We then apply threshold proposed by [22] to select the top-ranked features, which is based on a cumulative relevance. The number of features to keep is the one that represents 95% of the cumulative importance. Afterward, we use the N2 (intra-extra class separability) complexity measure in a multivariate setup to refine the feature subset. Another threshold is applied based on the number of features, as proposed in [10], in which datasets with a higher number of features get a larger cut.

Our results have shown that the data complexity measures can effectively select the most relevant features in a dataset by significantly reducing the feature space. On synthetic datasets, on average, we were able to achieve +6.14% increase in accuracy with a reduction of 98.89% on the number of features and on microarray datasets, an achievement of +3.07% increase in accuracy with a reduction of 99.65% in the number of features. These results show the robustness of our proposal. On classical datasets, we actually had a decrease of -7.33% in accuracy with a reduction of 90.01% of the dataset features. This decrease in accuracy can be attributed to the sharp reductions in the number of features and that removing around 90% of the features can eliminate a significant amount of relevant information for them. Therefore, better thresholds need to be defined in order to keep the most important features.

Another important contribution is which complexity measures are more suitable for FS. In a univariate setup, F1 was the most discriminative complexity measure. It presented robustness against noise and high-dimensionality. It is also one of the fastest complexity measures to be computed, with an asymptotic complexity of $O(d \cdot n)$. As already mentioned, this measure is largely applied to FS already. In a multivariate setup, N2 was the measure that could reflect most data simplicity when combined to F1. It is based on distance of the nearest neighbor of the same class (inter) and opposite class (intra) and, by doing so, it reflects the distribution and separability of the classes. It achieved a good performance in both forward and backward selection in the synthetic datasets.

5.2 LIMITATIONS

Despite of the good results, this work has some limitations. One could question the effectiveness of the method we used as the stopping criterion of FS. We focused on using tight thresholds to compensate the computational burden of the multivariate experiments. A multivariate analysis is a combinatorial problem that scales fast for a medium number of features. Therefore, we looked for solutions that would shrink the size of the feature space faster the others. Even though we were able to achieve good results in general, in datasets such as the case of LED500no, our threshold was detrimental to the accuracy results. In this dataset, there are seven relevant features but our method selected only five of them. This fact caused a loss of information that affected the predictive results achieved. This may also happen on the classical datasets employed. Another drawback is the use of a fixed threshold on multivariate analysis. We could have explored adaptive methods that consider complexity measure values ongoing along the process like the authors on [2].

All experiments and codes in this work were implemented in Python. We have used an ongoing development code of the complexity measures available at: `https://github.com/ricoms/gpam_stats`. However, it could also have been made in R, where there is a state-of-art implementation available at the ECol library (`https://github.com/lpfgarcia/ECoL`).

We could also have incorporated other data complexity measurements with this work. Since 2002, the use of data complexity measures is a growing field of research and many other measures have been developed. The measures presented in this paper demonstrated a difficulty with nonlinear datasets, as Parity3+3 and XOR-500. More sophisticated statistical measures could be used to get better insights of solutions in these cases.

Last but not least, we must investigate other forms of combining the complexity measures. In this work we have split the option by univariate and multivariate analysis. But one could also propose univariate analysis considering both F1 and N2 at the same time. One way of doing so would be to find a formula that would leverage the best characteristics of each complexity measure according to a given dataset. Some parameter would weight more one measure than another given a nonlinear characteristic or an increased addition of noise. This kind of solution would make the usage of complexity measures in FS more general and possibly more accurate.

5.3 FUTURE WORK

As already mentioned, future work should consider developing a better understanding of ways to combining the complexity measures in a way to explore the best characteristic of each one.

Determining a stopping criterion is also a not easy-to-solve-problem but it is critical to FS. Some good solutions are effective concerning runtime and others do not consider relevant information. Some sort of mix of both characteristics must be provided. Another possible solution is to apply heuristics to search for a subset of feature in less time. Works like Yusta [58] and Bermejo *et al.* [5] have shown great results.

APPENDIX

This appendix contains the complete results per dataset for all performance metrics and complexity measures.

A.1 UNIVARIATE RESULTS

Table 9: Univariate selection results for AUC.

Base	F1	F2	F3	F4	ReliefF
CorrAL-500	0.7484	0.2500	0.2500	0.2500	0.7495
Led500no	1.0000	1.0000	1.0000	0.0000	1.0000
Led500n15	1.0000	0.4285	0.4285	0.1428	0.8571
Monk1	0.3333	0.6667	0.6667	0.3333	0.3333
Monk2	0.1595	0.1667	0.1667	0.1667	0.0000
Monk3	0.6667	0.0000	0.0000	0.0000	0.6667
Parity3+3	0.0000	0.3333	0.3333	0.3333	0.0000
SD1	0.0000	0.5000	0.0000	0.0000	0.0000
SD2	0.4998	0.2500	0.4998	0.0000	0.0000
SD3	0.4998	0.0000	0.0000	0.0000	0.0000
XOR-500	0.0000	0.5000	0.5000	0.5000	0.0000

Table 10: Univariate selection results for Coverage.

Base	F1	F2	F3	F4	ReliefF
CorrAL-500	0.9900	0.3200	0.3200	0.3200	0.9880
Led500no	0.9860	0.9860	0.9860	0.0000	0.9860
Led500n15	0.9860	0.3180	0.3180	0.0000	0.9800
Monk1	0.2600	0.2700	0.2700	0.0000	0.3700
Monk2	0.0000	0.2800	0.2800	0.2800	0.0900
Monk3	0.5200	0.2800	0.2800	0.2800	0.2200
Parity3+3	0.0060	0.3200	0.3200	0.3200	0.3100
SD1	0.9967	0.9952	0.9965	0.0005	0.0027
SD2	0.9982	0.9982	0.9975	0.0000	0.0000
SD3	0.9975	0.9705	0.9875	0.0067	0.0000
XOR-500	0.2520	0.3200	0.3200	0.3200	0.5960

Table 11: Univariate selection results for Precision.

Base	F1	F2	F3	F4	ReliefF
CorrAL-500	0.7500	0.2500	0.2500	0.2500	0.7500
Led500no	1.0000	1.0000	1.0000	0.0000	1.0000
Led500n15	1.0000	0.4285	0.4285	0.1428	0.8571
Monk1	0.3333	0.6667	0.6667	0.3333	0.3333
Monk2	0.1667	0.1667	0.1667	0.1667	0.0000
Monk3	0.6667	0.0000	0.0000	0.0000	0.6667
Parity3+3	0.0000	0.3333	0.3333	0.3333	0.0000
SD1	0.0000	0.5000	0.0000	0.0000	0.0000
SD2	0.5000	0.2500	0.5000	0.0000	0.0000
SD3	0.5000	0.0000	0.0000	0.0000	0.0000
XOR-500	0.0000	0.5000	0.5000	0.5000	0.0000

Table 12: Backward selection results of AUC.

Base	N1	N2	ReliefF
CorrAL-500	0.5000	0.5000	0.5000
Led500no	0.4000	0.3000	0.0000
Led500n15	0.4000	0.3000	0.0000
Monk1	0.2500	0.2500	0.2500
Monk2	0.0000	0.0000	0.0000
Monk3	0.0000	0.0000	0.0000
Parity3+3	0.0000	0.0000	0.0000
SD1	0.4762	0.9500	0.9500
SD2	0.2386	0.2386	0.0000
SD3	0.3182	0.0000	0.0000
XOR-500	0.0000	0.0000	0.0000

Table 13: Backward selection results of Coverage.

Base	N1	N2	ReliefF
CorrAL-500	0.2000	0.0000	0.0000
Led500no	0.0000	0.0000	0.0000
Led500n15	0.0000	0.0000	0.0000
Monk1	0.0000	0.0000	0.0000
Monk2	0.0000	0.0000	0.0000
Monk3	0.0000	0.0000	0.0000
Parity3+3	0.0000	0.0000	0.0000
SD1	0.0000	0.9048	0.9048
SD2	0.0000	0.0000	0.0000
SD3	0.0000	0.0000	0.0000
XOR-500	0.0000	0.0000	0.0000

Table 14: Backward selection results of Precision.

Base	N1	N2	ReliefF
CorrAL-500	1.0000	0.7500	0.7500
Led500no	0.4286	0.2857	0.7143
Led500n15	0.5714	0.2857	0.7143
Monk1	0.3333	0.3333	0.3333
Monk2	0.0000	0.0000	0.0000
Monk3	0.6667	0.6667	0.6667
Parity3+3	0.0000	0.0000	0.0000
SD1	0.5000	1.0000	1.0000
SD2	0.2500	0.2500	0.0000
SD3	0.3333	0.0000	0.0000
XOR-500	0.0000	0.0000	0.0000

Table 15: Forward selection results of AUC.

Base	N1	N2	ReliefF
CorrAL-500	0.5000	0.5000	0.5000
Led500no	0.4000	0.0000	0.0000
Led500n15	0.4000	0.0000	0.0000
Monk1	0.2500	0.2500	0.2500
Monk2	0.0000	0.0000	0.0000
Monk3	0.0000	0.0000	0.0000
Parity3+3	0.0000	0.0000	0.0000
SD1	0.0000	0.9500	0.0000
SD2	0.7100	0.9500	0.0000
SD3	0.6300	0.9400	0.0000
XOR-500	0.0000	0.0000	0.0000

Table 16: Forward selection results of Coverage.

Base	N1	N2	ReliefF
CorrAL-500	0.0000	0.2000	0.0000
Led500no	0.0000	0.0000	0.0000
Led500n15	0.0000	0.0000	0.0000
Monk1	0.0000	0.0000	0.0000
Monk2	0.0000	0.0000	0.0000
Monk3	0.0000	0.0000	0.0000
Parity3+3	0.0000	0.0000	0.0000
SD1	0.0000	0.9000	0.0000
SD2	0.0000	0.8200	0.0000
SD3	0.0000	0.7400	0.0000
XOR-500	0.0000	0.0000	0.0000

Table 17: Forward selection results of Precision.

Base	N1	N2	ReliefF
CorrAL-500	0.7500	1.0000	0.7500
Led500no	0.4300	0.7100	0.7100
Led500n15	0.4300	0.7100	0.7100
Monk1	0.3300	0.3300	0.3300
Monk2	0.0000	0.0000	0.0000
Monk3	0.6700	0.6700	0.6700
Parity3+3	0.0000	0.0000	0.0000
SD1	0.0000	1.0000	0.0000
SD2	0.7500	1.0000	0.0000
SD3	0.6700	1.0000	0.0000
XOR-500	0.0000	0.0000	0.0000

BIBLIOGRAPHY

[1] Almuallim, H.; Dietterich, T. G. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, v. 69, n. 1-2, p. 279–305, 1994.

[2] B. Seijo-Pardo, V. Bolon-Canedo, A. A.-B. On developing an automatic threshold applied to feature selection ensembles. *Information Fusion*, v. 45, p. 227–245, 2018.

[3] Baumgartner, R.; Ho, T. K.; Somorjai, R.; Himmelreich, U.; Sorrell, T. Complexity of magnetic resonance spectrum classification. In: *Data Complexity in Pattern Recognition*, Springer, p. 241–248, 2006.

[4] Belanche, L. A.; González, F. F. Review and evaluation of feature selection algorithms in synthetic problems. *arXiv preprint arXiv:1101.2320*, 2011.

[5] Bermejo, P.; Gámez, J. A.; Puerta, J. M. A grasp algorithm for fast hybrid (filter-wrapper) feature subset selection in high-dimensional datasets. *Pattern Recognition Letters*, v. 32, n. 5, p. 701–711, 2011.

[6] Berton, L.; de Paulo Faleiros, T.; Valejo, A.; Valverde-Rebaza, J.; de Andrade Lopes, A. Rgcli: Robust graph that considers labeled instances for semi-supervised learning. *Neurocomputing*, v. 226, p. 238–248, 2017.

[7] Bhanot, G.; Alexe, G.; Venkataraghavan, B.; Levine, A. J. A robust meta-classification strategy for cancer detection from ms data. *Proteomics*, v. 6, n. 2, p. 592–604, 2006.

[8] Blum, A. L.; Langley, P. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, v. 97, n. 1, p. 245–271, 1997.

[9] Bolón Canedo, V. Novel feature selection methods for high dimensional data. 2014.

[10] Bolón-Canedo, V.; Sánchez-Maroño, N.; Alonso-Betanzos, A. A review of feature selection methods on synthetic data. *Knowledge and Information Systems*, v. 34, n. 3, p. 483–519, 2013.

[11] Bolón-Canedo, V.; Sánchez-Maroño, N.; Alonso-Betanzos, A. A distributed feature selection approach based on a complexity measure. In: *International Work-Conference on Artificial Neural Networks*, Springer, 2015, p. 15–28.

[12] Bolón-Canedo, V.; Sánchez-Marono, N.; Alonso-Betanzos, A.; Benítez, J. M.; Herrera, F. A review of microarray datasets and applied feature selection methods. *Information Sciences*, v. 282, p. 111–135, 2014.

[13] Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Computers and Electrical Engineering*, v. 40, n. 1, p. 16–28, 2014.

[14] Chapelle, O.; Scholkopf, B.; Zien, A. Semi-supervised learning. *IEEE Transactions on Neural Networks*, v. 20, n. 3, p. 542–542, 2009.

[15] Cohen, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, v. 20, n. 1, p. 37–46, 1960.

[16] Cover, T. M.; Thomas, J. A. Elements of information theory 2nd edition. 2006.

[17] Dheeru, D.; Karra Taniskidou, E. UCI machine learning repository. 2017. Disponível em http://archive.ics.uci.edu/ml

[18] Ding, C.; Peng, H. Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, v. 3, n. 02, p. 185–205, 2005.

[19] Doak, J. An evaluation of feature selection methods and their application to computer security. *UC Davis Dept of Computer Science tech reports*, 1992.

[20] Dong, M.; Kothari, R. P. Feature subset selection using a new definition of classificability. *Pattern Recognition Letters*, v. 24, p. 1215–1225, 2003.

[21] Faceli, K.; Lorena, A. C.; Gama, J.; Carvalho, A. Inteligência artificial: Uma abordagem de aprendizado de máquina. *Rio de Janeiro: LTC*, v. 2, p. 192, 2011.

[22] Ferreira, A. J.; Figueiredo, M. A. Efficient feature selection filters for high-dimensional data. *Pattern Recognition Letters*, v. 33, n. 13, p. 1794–1804, 2012.

[23] Gheyas, I. A.; Smith, L. S. Feature subset selection in large dimensionality domains. *Pattern Recognition*, v. 43, n. 1, p. 5–13, 2010.

[24] Guyon, I.; Elisseeff, A. An introduction to variable and feature selection. *Journal of Machine Learning Research*, v. 3, n. Mar, p. 1157–1182, 2003.

[25] Guyon, I.; Weston, J.; Barnhill, S.; Vapnik, V. Gene selection for cancer classification using support vector machines. *Machine Learning*, v. 46, n. 1, p. 389–422, 2002.

[26] Hall, M. A. Correlation-based feature selection for machine learning. 1999.

[27] Hall, M. A.; Smith, L. A. Practical feature subset selection for machine learning. 1998.

[28] He, H.; Garcia, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, , n. 9, p. 1263–1284, 2008.

[29] He, X.; Cai, D.; Niyogi, P. Laplacian score for feature selection. In: *Advances in Neural Information Processing Systems*, 2006, p. 507–514.

[30] Ho, T. K.; Basu, M. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 3, p. 289–300, 2002.

[31] Hoekstra, A.; Duin, R. P. W. On the nonlinearity of pattern classifiers. In: *13th International Conference on Pattern Recognition (ICPR)*, 1996, p. 271–275.

[32] Hu, Q.; Pedrycz, W.; Yu, D.; Lang, J. Selecting discrete and continuous features based on neighborhood decision error minimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, v. 40, n. 1, p. 137–150, 2010.

[33] Jensen, C. A.; El-Sharkawi, M. A.; Marks, R. J. Power system security assessment using neural networks: feature selection using fisher discrimination. *IEEE Transactions on Power Systems*, v. 16, n. 4, p. 757–763, 2001.

[34] Jolliffe, I. *Principal component analysis*. Springer Series in Statistics. Springer New York, 2006.

[35] Kira, K.; Rendell, L. A. The feature selection problem: Traditional methods and a new algorithm. In: *AAAI Proceedings of the 10th National Conference on Artificial Intelligence*, 1992, p. 129–134.

[36] Kohavi, R.; John, G. H. Wrappers for feature subset selection. *Artificial Intelligence*, v. 97, n. 1-2, p. 273–324, 1997.

[37] Liu, H.; Motoda, H. *Feature selection for knowledge discovery and data mining*. The Springer International Series in Engineering and Computer Science. Springer US, 1998.

[38] Macià, N. *Data complexity in supervised learning: a far-reaching implication*. Ph.D Thesis, La Salle, Universitat Ramon Llull, 2011.

[39] Mejía-Lavalle, M.; Sucar, E.; Arroyo, G. Feature selection with a perceptron neural net. In: *Proceedings of the International Workshop on Feature Selection for Data Mining*, 2006, p. 131–135.

[40] Ming, L.; Vitanyi, P. *An introduction to Kolmogorov complexity and its applications*. Springer, 1993.

[41] Mitchell, T. M. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, v. 45, n. 37, p. 870–877, 1997.

[42] Mitra, P.; Murthy, C.; Pal, S. K. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 3, p. 301–312, 2002.

[43] Okimoto, L. C.; Savii, R. M.; Lorena, A. C. Complexity measures effectiveness in feature selection. In: *IEEE Proceedings of the Brazilian Conference on Intelligent Systems*, 2017.

[44] Pareto, V. *Cours d'économie politique*, v. 1. Librairie Droz, 1964.

[45] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

[46] Pranckeviciene, E.; Ho, T. K.; Somorjai, R. Class separability in spaces reduced by feature selection. In: *Proceedings of the 18th International Conference in Pattern Recognition*, 2006, p. 254–257.

[47] Quinlan, J. R. Induction of decision trees. *Machine Learning*, v. 1, n. 1, p. 81–106, 1986.

[48] Saeys, Y.; Inza, I.; Larrañaga, P. A review of feature selection techniques in bioinformatics. *Bioinformatics*, v. 23, n. 19, p. 2507–2517, 2007.

[49] Schölkopf, B.; Smola, A. J.; *et al. Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[50] Seijo-Pardo, B.; Bolón-Canedo, V.; Alonso-Betanzos, A. Using data complexity measures for thresholding in feature selection rankers. In: *Conference of the Spanish Association for Artificial Intelligence*, Springer, 2016, p. 121–131.

[51] Singh, S. Prism: A novel framework for pattern recognition. *Pattern Analysis and Applications*, v. 6, n. 2, p. 134–149, 2003.

[52] Skrypnyk, I. Irrelevant features, class separability, and complexity of classification problems. In: *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, IEEE, 2011, p. 998–1003.

[53] Tang, J.; Alelyani, S.; Liu, H. Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, p. 37, 2014.

[54] Vapnik, V. N.; Vapnik, V. *Statistical learning theory*, v. 1. Wiley New York, 1998.

[55] Witten, I. H.; Frank, E.; Hall, M. A.; Pal, C. J. *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[56] Wolpert, D. H. The lack of a priori distinctions between learning algorithms. *Neural Computation*, v. 8, n. 7, p. 1341–1390, 1996.

[57] Yu, L.; Liu, H. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, v. 5, n. Oct, p. 1205–1224, 2004.

[58] Yusta, S. C. Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letters*, v. 30, n. 5, p. 525–534, 2009.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of August 22, 2018 (`classicthesis` version 1.0).

DECLARATION

Put your declaration here.

São José dos Campos, July 2018

Lucas Chesini Okimoto